

HAKING

PRACTICAL PROTECTION

IT SECURITY MAGAZINE



GUIDE TO NMAP THE SECOND ENCOUNTER

NMAP: THE RIGHT TOOL FOR THE JOB

HOW TO USE THE SWISS ARMY KNIFE
OF HACKING AND NOT GET CUT?

NMAP - HOLLYWOOD'S
HACKING TOOL OF CHOICE

NMAP KUNG-FU

Vol.8 No.04
Issue 04/2013(64) ISSN: 1733-7186

PLUS

IN DEPTH GUIDE
TO DIGITAL FORENSICS



Dr.Web SpIDer is 8-legged!



New Version 8.0

Security Space and Dr.Web Antivirus for Windows

Get your free 60-day license under <https://www.drweb.com/press/> to protect your PC and your smartphone with Dr.Web!

Your promo code: **Hakin9**

Protect your mobile device free of charge!

https://support.drweb.com/free_mobile/



Atola Insight

That's all you need for data recovery.

Atola Technology offers *Atola Insight* – the only data recovery device that covers the entire data recovery process: *in-depth* **HDD diagnostics**, **firmware recovery**, **HDD duplication**, and **file recovery**. It is like a whole data recovery Lab in one Tool.

This product is the best choice for seasoned professionals as well as start-up data recovery companies.

Emphasized features at a glance:

- Automatic in-depth diagnostic of all hard drive components
- Automatic firmware recovery and ATA password removal
- Very fast imaging of damaged drives
- Imaging by heads
- Case management
- Real time current monitor
- Firmware area backup system
- Serial port and power control
- Write protection switch



Visit atola.com for details



HAKIN9 team

Editor in Chief: Krzysztof Krokwa
krzysztof.krokwa@software.com.pl

Editorial Advisory Board: John Webb, Marco Hermans, Gareth Watters, Kishore P.V.

Proofreaders: Krzysztof Krokwa, Krzysztof Samborski

Special Thanks to the Beta testers and Proofreaders who helped us with this issue. Without their assistance there would not be a Hakin9 magazine.

Senior Consultant/Publisher: Paweł Marciniak

CEO: Ewa Dudzic
ewa.dudzic@hakin9.org

Product Manager: Krzysztof Samborski
krzysztof.samborski@hakin9.org

Production Director: Andrzej Kuca
andrzej.kuca@hakin9.org

Marketing Director: Krzysztof Krokwa
krzysztof.krokwa@software.com.pl

DTP: Ireneusz Pogroszewski
Art Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl

Publisher: Hakin9 Media sp. z o.o. SK
02-676 Warszawa, ul. Postępu 17D
Phone: 1 917 338 3631
www.hakin9.org

Whilst every effort has been made to ensure the highest quality of the magazine, the editors make no warranty, expressed or implied, concerning the results of the content's usage. All trademarks presented in the magazine were used for informative purposes only.

All rights to trade marks presented in the magazine are reserved by the companies which own them.

DISCLAIMER!

The techniques described in our articles may only be used in private, local networks. The editors hold no responsibility for misuse of the presented techniques or consequent data loss.

Dear Readers,

Welcome to this very special issue of Hakin9. For the second time we will be touching a very controversial subject – scanning with nmap.

Last year we published an issue on nmap which made the whole Internet boil and this time we're going to do it again. We are going to surprise everyone with plethora of fascinating content that will make your head spin.

In this month's edition, Aamir Lakhani, Andrew Brooker, Daniel Renaud, Andrey Moskvitin, Nathan Swaim and Justin Hutchens will show you why nmap is The Right Tool for job. Andrew Jones, Evan Francen, Jake Wylezek, James Tan and Joshua Cornutt will teach how to use The Swiss Army Knife of Network Discovery and how not to cut yourself with it, while Branden Paul, Sergio Castro, Tony Lee and Peter Harmsen are going to show you few advanced tricks.

Special mention goes to Antonio Ierano with his outstanding In Depth Guide To Digital Forensic.

Hakin9's Editorial Team would like to give special thanks to the authors, betatesters and proofreaders.

We hope our effort was worthwhile and the Haking Extra's BackTrack 5r3 issue will appeal to you. We wish you a nice read.

Most special thanks goes to Gordon „Fyodor” Lyon for creating such amazing, open source tool which to this day still is a bread and butter tool for both hackers and IT security professionals.

Krzysztof Krokwa
Editor of Hakin9 team.

THE RIGHT TOOL

NMAP Kung-Fu **6**

*By Aamir Lakhani,
DCUCD, DCUCI, CCNP, CCDP, Microsoft Certified Systems Engineer, IBM Cloud Computing Architect, CISSP, HP Open View Professional*

Map and Network **12**

*By Andrew Brooker
CISSP, CRISC Director of Operations Assurity River Group*

The Bread and Butter of IT Security **16**

*By Andrey Moskvitin
IT Security Professional, Microsoft*

NMAP Scanning: How a Simple Tool STILL Makes Dramatic Impact **20**

*By Nathan Swaim
President, ANRC*

Nmap: a “Hacker Tool” for Security Professionals **26**

CISSP, CEH, ECSA, CHFI

THE SWISS ARMY KNIFE

Nmap: For Newbies **36**

*By Andrew Jones
VMTraining GSEC, GCIH, CVE5, VMTraining Certified Trainer*

Nmap – The Tool of Almost Endless Capabilities **40**

*By Evan Francen
President, FRSecure LLC & Information Security Evangelist CISSP, CISM, CCSK*

NMAP – Get To Know the Network **44**

*By Jake Wylezek
Solutions/Systems Engineer at Hewlett-Packard*

Nmap – The Swiss Army Knife of Network Discovery **50**

*By James Tan
BSc Psychology, ISO 27001, CISSP, CCSK, CI-SA, eCPPT, PMP*

Practical Nmap Scanning **56**

*By Joshua Cornutt
CompTIA A+ Certified Professional IT Technician*

ADVANCED APPROACH

Nmap – The Multitool of Network Discovery **60**

*By Branden Paul
Network Administrator, Banking Company*

Using Nmap for Outbound Traffic Analysis **64**

*By Sergio Castro
Managing Director, Qualys Latin America*

Refining Your Nmap Scan Strategy **68**

*By Tony Lee
Principal security consultant at FireEye*

Install and Configure A Working Port Scan Attack Daemon PSAD on Ubuntu **76**

*Peter Harmsen
MCSA*

Introduction to Nmap as a Computer Forensics Tool **86**

*By Antonio Ierano
Former Cisco European Security Evangelist & Senior Consultant*

NMAP Kung-Fu

Nmap is a popular tool for network reconnaissance. It is usually one of the first tools a network penetration tester will use to determine the type of system they are targeting, what ports are open on the target system, and what services may be running on the system.

Nmap stands for “network mapper” and is used to scan hosts and services on a network. Nmap has advanced features that can detect different applications running on systems as well as services and OS fingerprinting features.

The term “Reconnaissance,” by definition comes from the military warfare strategy of exploring beyond the area occupied by friendly forces to gain information about the enemy for future analysis or attack. Reconnaissance of computer systems is similar in nature meaning typically a penetration tester or hacker will attempt to learn as much as possible about a target’s environment and system traits prior to launching an attack. This is also known as establishing a “Footprint” of a target.

Reconnaissance is typically passive and in many cases not illegal to do as long as you don’t complete a three way handshake with an unauthorized system.

Assuming a party doesn’t send an ACK back once a system responds with an ACK after a SYN-ACK has been established, the effort is legal, but, for example, scanning with Nmap using `-sT` is illegal while using `-sS` is not illegal based on this principle.

Confusing? Don’t worry, we will get into the options for Nmap and how to use them.

Because of the popularity of Nmap, many security vendors such as IPS/IDS and Firewall ven-

dors will be easily able to detect Nmap scans with Nmap’s default settings. In most cases, I find that this does not often matter because rarely are the logs reviewed, but if they are then a key advantage of the penetration tester is lost – stealth.

Using Nmap

Nmap is available on almost all operating systems. It can be downloaded and installed on Windows, OS X, Linux, and even jailbroken and rooted mobile devices.

Installing Nmap is pretty simple. On most Debian based Linux systems you can open up a terminal window and type in the command:

```
sudo apt-get install nmap
```

Many systems come with Nmap preinstalled; therefore you can just start using the program.

For the purposes of this article we will assume Nmap on Backtrack 5 RC3 is being used. Nmap comes preinstalled on Backtrack 5 so there is no need to install it.

The commands and basic usage for Nmap are relatively the same regardless of what platform you use it on.

Let’s get started. To use Nmap we will bring up a terminal window (command prompt if you installed this on Windows).

In it's most basic syntax Nmap can be simply executed by typing `nmap target_ip_address` (Figure 1). We can see Nmap scanned a few common ports and reported on what ports were open. This is the most common type of scan. Nmap by default will try and use reverse DNS to resolve names to IP addresses. In some cases you might want to disable DNS reverse lookup. The reason I personally like to disable it is because it usually speeds up the process tremendously once reverse DNS has been disabled. You can see the exact same scan with DNS reverse lookup turned off completed in less than 1 second versus 13 seconds when the DNS reverse lookup was enabled (Figure 2).

Scanning with Nmap

Typically as a penetration tester you might want to get a quick look of all the devices on your network. You can use Nmap to perform a ping sweep. Nmap will report on any hosts that respond back with an ICMP echo-reply. We do this by issuing the command: `nmap -sP network/mask` (Figure 3).

As you can see, Nmap was able to see a few different devices. However, not all devices will respond to ICMP. Many security devices are setup to drop ICMP traffic.

```

root@bt:~
File Edit View Terminal Help

SIOCSIFADDR: No such device
wlan0: ERROR while getting interface flags: No such device
wlan0: ERROR while getting interface flags: No such device
Bind socket to interface: No such device
Failed to bring up wlan0.

root@bt:~#
root@bt:~# nmap 10.0.1.238

Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-08 23:04 CDT
Nmap scan report for 10.0.1.238
Host is up (0.00017s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
5000/tcp  open  upnp
MAC Address: 08:0C:29:33:FD:70 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 13.14 seconds
root@bt:~#
    
```

Figure 1. Basic Nmap Scanning Technique

```

SIOCSIFADDR: No such device
wlan0: ERROR while getting interface flags: No such device
wlan0: ERROR while getting interface flags: No such device
Bind socket to interface: No such device
Failed to bring up wlan0.

root@bt:~#
root@bt:~# nmap 10.0.1.238

Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-08 23:04 CDT
Nmap scan report for 10.0.1.238
Host is up (0.00017s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
5000/tcp  open  upnp
MAC Address: 08:0C:29:33:FD:70 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 13.14 seconds
root@bt:~#
    
```

Figure 2. Disabling Reverse DNS Scan Technique

A common technique, which penetration testers use to find devices that may drop ICMP is to scan for a particular port. If the port accepts connections, then Nmap will report it as a live host. In the following example we will scan for port 80 open ports on our network. To scan a network for a particular port we will issue the following command (Figure 4):

```
nmap -pT80 10.0.1.238
```

Notice, this time we got slightly different results, which are because we are looking for devices that accept a connection on port 80, or in other words devices that are most likely running web servers.

It is also possible to do an OS fingerprint with Nmap, as it will attempt to find out what the target's operating system and installed services are when you conduct this scan. You simply use the `-o` option to start this scan. We could run the same command again and add the `-o` option.

```
nmap -pT80 10.0.1.238
```

Nmap Stealth Scans

The popularity of Nmap makes it very detectable. Therefore, many penetration testers will find the

```

Nmap done: 256 IP addresses (12 hosts up) scanned in 3.03 seconds
root@bt:~# clear
root@bt:~# nmap -sP 10.0.1.0/24

Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-09 09:42 CDT
Nmap scan report for 10.0.1.1
Host is up (0.0014s latency).
MAC Address: 00:10:04:43:43:05 (Meraki)
Nmap scan report for 10.0.1.50
Host is up (0.023s latency).
MAC Address: 70:56:01:F3:79:08 (Unknown)
Nmap scan report for 10.0.1.107
Host is up (0.044s latency).
MAC Address: 00:9C:02:C7:76:1D (Hewlett-Packard Company)
Nmap scan report for 10.0.1.159
Host is up.
Nmap scan report for 10.0.1.238
Host is up (0.00023s latency).
MAC Address: 00:0C:29:33:FD:70 (VMware)
Nmap scan report for 10.0.1.240
Host is up (0.00034s latency).
MAC Address: B8:F6:B1:11:9E:A1 (Apple)
Nmap scan report for 10.0.1.244
Host is up (0.012s latency).
MAC Address: 00:00:05:00:00:0F (EchoStar Global B.V.)
Nmap scan report for 10.0.1.245
Host is up (0.016s latency).
MAC Address: 9C:20:7B:93:00:73 (Unknown)
Nmap scan report for 10.0.1.248
Host is up (0.060s latency).
MAC Address: 88:53:95:84:4F:44 (Unknown)
    
```

Figure 3. Nmap echo-reply Scanning Technique

```

root@bt:~# nmap PT80 10.0.1.238

Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-09 11:27 CDT
Nmap scan report for PT80 (198.153.194.3)
Host is up (0.028s latency).
Not shown: 998 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   closed https

Nmap scan report for 10.0.1.238
Host is up (0.00016s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
5000/tcp  open  upnp
MAC Address: 08:0C:29:33:FD:70 (VMware)

Nmap done: 2 IP addresses (2 hosts up) scanned in 5.50 seconds
root@bt:~#
    
```

Figure 4. Nmap Port Scanning

need to perform stealth scans or spoof their IPs. When an attacker spoofs an IP, they are disguising their scans coming from a different IP address. They are hoping that if they mimic a trusted server their scans will go unnoticed. In many cases I will pick a network management server or another “chatty” server that has legitimate needs to communicate with multiple network devices.

To spoof an IP address, use the option `-D` followed by the IP address you want to appear as. The `-D` switch means you will use a decoy.

We will now scan our a specific host on our network (10.0.1.238) and make it appear as if the scans are originating from 10.0.1.1.

We issue the following command (Figure 5):

```
nmap 10.0.1.238 -D 10.0.1.1
```

Nmap Output Results

You can make Nmap write it's results out to a file to examine offline. You simply need to add the `-oN filename.txt` command at the end of your Nmap command. In our case we typed the following command:

```
nmap -sP -n -pT80 10.0.1.0/24 -oN scan1.txt
```



Figure 5. Nmap Stealth Scan

Our results were saved to a text file `scan1.txt` that we can open up in any text editor

Using Zenmap with Nmap

Additionally, many people use Nmap with Zenmap. Zenmap is a GUI front end to Nmap. Zenmap gives nmap a graphical user interface to run commands.

Although there are many purist who will tell you the command-line version is the best version because of its speed and flexibility, Zenmap has come a long way and has almost Nmap features. Zenmap also offers exclusive features not offered in Nmap such as developing graphical representations of a scan which can be used later by other reporting systems.

Zenmap also allows you to choose options for your scans via the GUI instead of remembering the individual Nmap options and switches.

Zenmap is usually installed when you install Nmap on your system. To open up Zenmap, go to the Backtrack menu. Go to Network Mapping, port mapping, and launch Zenmap (Figure 6).

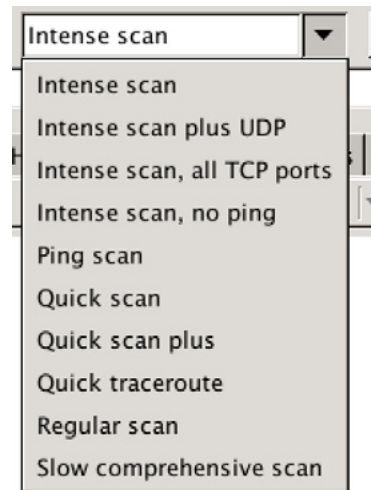


Figure 7. Zenmap Scan Menu

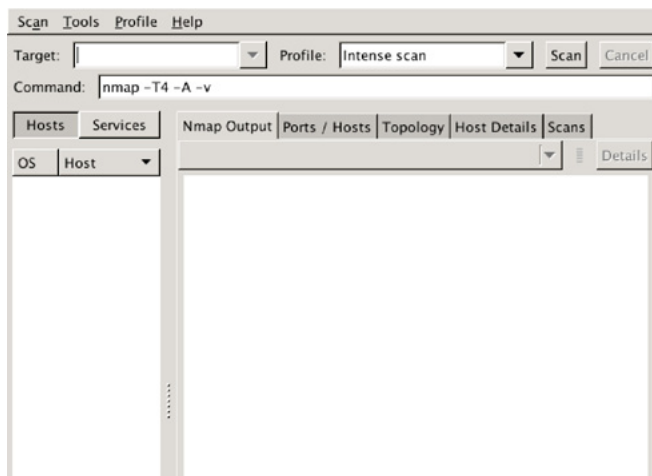


Figure 6. Zenmap Launch Screen

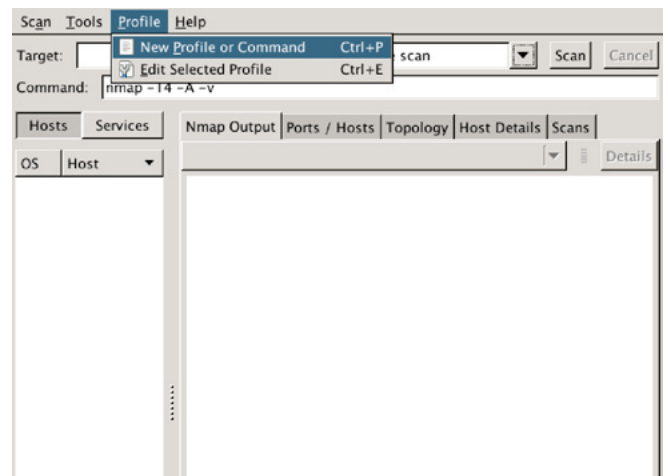


Figure 8. Creating a profile with Zenmap

You will notice under the scan menu that there are several options to determine what type of scan you would like to run (Figure 7).

The first step to create a new profile. A profile in Zenmap allows a penetration tester to create what type of scan to execute and what different options to include. Go to the profile menu and select new profile to create a new profile (Figure 8).

When you select new profile, the profile editor will launch. You will need to give your profile a descriptive name. For example, you can call the profile "My First Scan" or anything else you would like.

Optionally you can give the profile a description. During the course of using Zenmap you will probably create many profiles and make multiple scans. A natural reflex may be to delete profiles post execution. Here is a word of advice. Don't. Profiles don't take any space and come handy when you want to recreate something. Practice being extremely descriptive in profile names and come up with a standard naming method. I start all my profile descriptions with the date, time, description of my location, my target network scan location, and customer name (Figure 9).

When you have completed your description, click on the scan tab. In the target section you will add what hosts or networks you would like to scan. This field can take a range of IP addresses

(10.0.1.1-255) or it can take a network in CIDR format (10.0.1.0/24).

You can see option `-A` is selected by default to enable aggressive scanning. Aggressive scanning will enable OS detection (`-O`), version scanning (`-sV`), script scanning (`-sC`) and traceroute (`--traceroute`). Essentially aggressive scanning allows a user to turn on multiple flags without the need of having to remember them.

Aggressive scanning is considered intrusive, which means most security devices will detect it. An aggressive scan may go unnoticed if the target is an extremely specific host, but regardless of the situation, it's recommended you have permission to scan before using this option. As a reminder, completing the ACK in the three-way handshake with a unauthorized system is considered illegal by US standards.

Information received from DNS reconnaissance exercises can be used to target a very specific host. Before we do that, lets set a few common options first (Figure 10).

Select the ping tab. Select the `-Pn` flag option so nmap will not ping the host first. When this flag is not set, nmap will ping your target hosts and networks. Default settings only perform scans on hosts that are considered alive or reachable. `-Pn` flag tells nmap to scan a host even without a ping response. Although this makes the scan considerably more lengthy, the `-Pn` flag allows nmap to avoid a common problem of not receiving a ping response when the ping requests are blocked by security defenses (Figure 11).

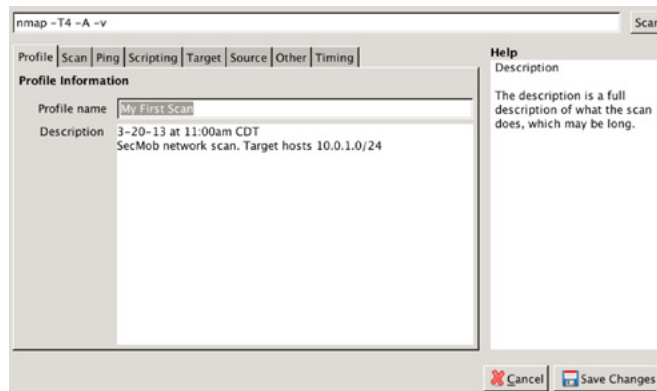


Figure 9. Zenmap Scanning Profile Configuration

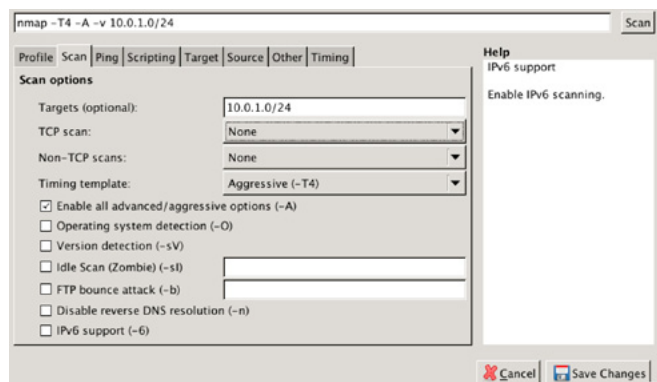


Figure 10. Aggressive Scanning Options

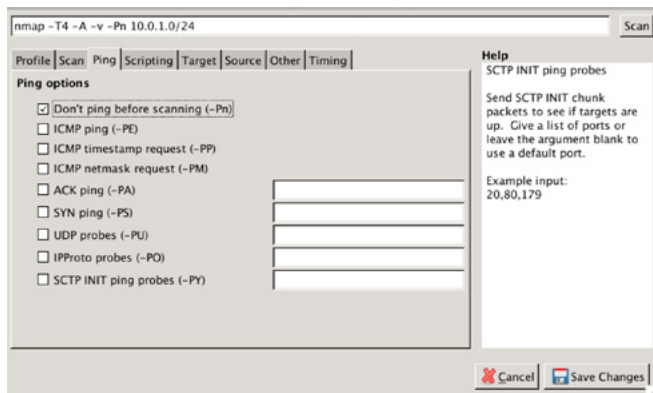


Figure 11. ICMP Options

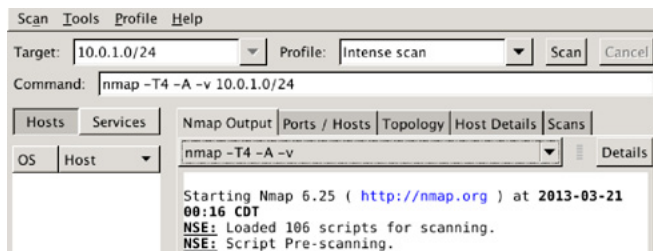


Figure 12. Nmap Output

Save changes made by selecting the save button in the lower right hand corner. Once saved, select the scan button on the top right side of the screen to start the scan. Notice that your options and the target that you configured in the profile editor are listed (Figure 12).



World Wide Technology, Inc.

World Wide Technology, Inc. (WWT) is a leading Systems Integrator providing technology products, services, and supply chain solutions to customers around the globe. WWT's IT security practice delivers a principled approach to achieving holistic security posture by focusing on prevention, detection and remediation, to create operational readiness and tactical security architectures that align with an organization's strategic areas of concern. www.wwt.com.

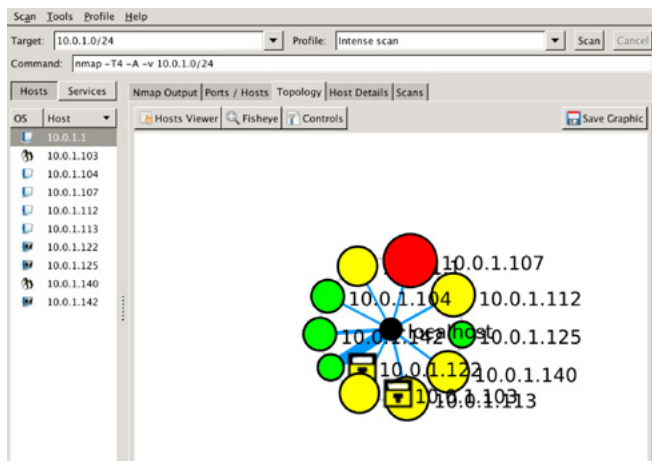


Figure 13. Network Topology

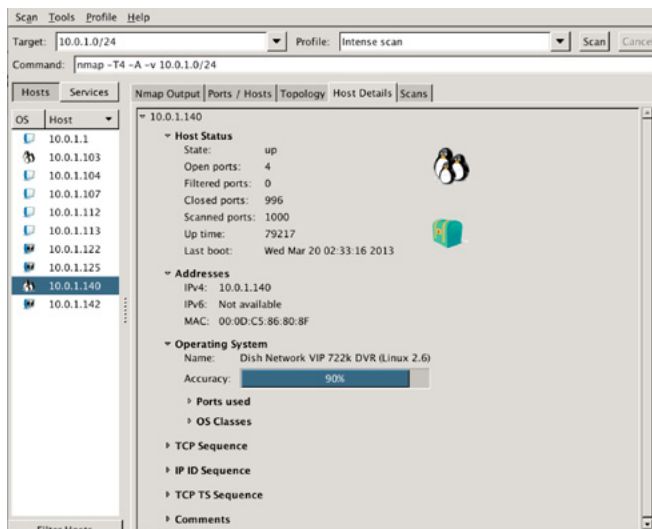


Figure 14. Network Host Details

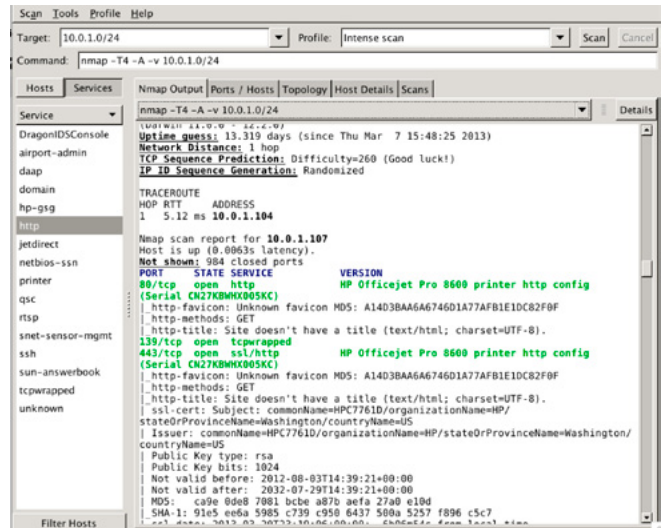


Figure 15. Nmap full output

The network topology tab will give you a quick look at how your scan on the target network was completed and if you had to cross any routers. In this example, you see the scan stayed local to the network (Figure 13). The host tab will give a list of the hosts discovered. When a host is selected, Zenmap will display a detailed list of the host, its operating system, and common services. In this example you can see one of our hosts is a satellite DVR/receiver combo (Figure 14). If you look at the scan window you will not only see what ports are open on specific hosts, but also what applications are running on those hosts. Notice that Nmap can determine things such as, whether a server is running IIS 5.0 as a web server over port 80. You now know the server, the operating system, and the web applications a particular host is running which is extremely valuable for the next step in a penetration testing exercise (Figure 15). It is now possible for you to concentrate your efforts on the target's running web services over port 80 since it is open.

AAMIR LAKHANI



Aamir Lakhani is a leading Cyber Security and Cyber Counter Intelligence architect and researcher for World Wide Technology, Inc (WWT). Aamir Lakhani is responsible for WWT's efforts to provide IT security solutions to major commercial and federal enterprise organizations. He has worked

on large projects for the many of the Fortune 500, US Department of Defense, major healthcare, educational providers, global financial institutions, and large media companies. He is well known for as one of the leading offensive security and offensive counter measures specialist. You can follow Aamir Lakhani on Twitter @ aamirlakhani and on his blog at www.DrChaos.Com.

Secure User Authentication

Image-Based Authentication that Generates One-Time Passwords

- Strong user authentication that's easy to use.
- A flexible layer of authentication that can be inserted anywhere needed.
- Generates one-time passwords for strong security.
- Protect user accounts, prevent fraud and stolen credentials.
- For websites, mobile apps, or two-factor authentication.



Get It Now!

Visit www.ConfidentTechnologies.com to get APIs or start a free trial today.







Confident[®]
TECHNOLOGIES

Login:

Username:

Password:

Identify your secret categories to form a one-time password:

Tap your secret categories to authenticate:

Map and Network

Network Mapper is a network scanner that is used to discover network hosts and their services. The initial driver for Gordon Lyon was to create a utility that could “map the network”, hence nmap. Back in 1997, nmap was a Linux only utility, but today it is a cross-platform, lightweight network security scanner. Not only can you use nmap on your favorite OS, but you have the option between CLI or GUI.

Let's get started: I'm using Ubuntu 12.04 LTS. To install nmap from a terminal, run: `sudo apt-get install nmap` (Figure 1).

Now, let's talk about targeting. Nmap does both IPv4 and IPv6. With IPv4 you can use a variety of notation to scan entire subnets, specific addresses, or spaces of a subnet; You can even scan multiple subnets in one command or point to a list of targets in a file.

Without any options, just with a target address, let's see what happens: `nmap <ip.add.res.s>` (Figure 2).

As you can see there are several open ports/services running on this host. You will see that “Not Shown” are 990 ports where an application/service didn't respond. However, the interesting part is the

10 open ports/services that did respond. So in less than ½ a second, we learned there is a web server (HTTP/S), network file system server (NFS), and common internet file system server (CIFS)(tcp/445).

By default, without options, nmap will scan (check) the top 1000 most used ports. This list of ports is based on Internet research, so while this may represent 93%-95% of open ports on the Internet, it may not be statistically accurate for internal networks. Also, these top 1000 ports are a mixture of protocols and numbers. Generally speaking, ports 1-1024 are the most “well-known” but as time

```
andrewb@HURRICANE:~$ sudo apt-get install nmap
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  nmap
0 upgraded, 1 newly installed, 0 to remove and 30 not upgraded.
Need to get 1,643 kB of archives.
After this operation, 6,913 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu/ precise/main nmap amd64 5.21-1.1ubuntu1 [1,643 kB]
Fetched 1,643 kB in 11s (149 kB/s)
Selecting previously unselected package nmap.
(Reading database ... 312817 files and directories currently installed.)
Unpacking nmap (from .../nmap_5.21-1.1ubuntu1_amd64.deb) ...
Processing triggers for man-db ...
Setting up nmap (5.21-1.1ubuntu1) ...
andrewb@HURRICANE:~$
```

Figure 1. Interface overview

```
andrewb@HURRICANE:~$ nmap 192.168.1.5
Starting Nmap 5.21 ( http://nmap.org ) at 2013-05-08 21:59 CDT
Nmap scan report for 192.168.1.5
Host is up (0.0060s latency).
Not shown: 990 closed ports
PORT      STATE SERVICE
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
548/tcp   open  afp
2049/tcp  open  nfs
3689/tcp  open  rendezvous
9000/tcp  open  cslistener
49152/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.21 seconds
andrewb@HURRICANE:~$
```

Figure 2. Basics

goes on, there are more and more services that are widely used outside of this “well-known” range.

I bring up these nuances to help illustrate the importance of understanding your goal when executing an nmap scan. If you are simply trying to discover live hosts or if you are trying to inventory every service – there are certainly areas you may not touch depending on your options when executing a scan.

```
andrew@HURRICANE:~$ nmap 192.168.1.0/24 -p 80,443
Starting Nmap 5.21 ( http://nmap.org ) at 2013-05-08 23:22 CDT
Nmap scan report for 192.168.1.1
Host is up (0.0077s latency).
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   closed https

Nmap scan report for 192.168.1.5
Host is up (0.0086s latency).
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https

Nmap scan report for 192.168.1.6
Host is up (0.0082s latency).
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https

Nmap scan report for 192.168.1.101
Host is up (0.043s latency).
PORT      STATE SERVICE
80/tcp    closed http
443/tcp   closed https

Nmap scan report for 192.168.1.108
Host is up (0.0011s latency).
PORT      STATE SERVICE
80/tcp    closed http
443/tcp   closed https

Nmap scan report for 192.168.1.113
Host is up (0.016s latency).
PORT      STATE SERVICE
80/tcp    closed http
443/tcp   closed https

Nmap done: 256 IP addresses (6 hosts up) scanned in 3.49 seconds
andrew@HURRICANE:~$
```

Figure 3. Unexpected results

Let’s try another scan. This time let’s scan the network and look for web services. My network is a /24 in CIDR notation or in other words, addresses ranging from 0 to 255, 256 in total. Typically, your first address, 0 in this case, is often the “network identifier”

```
andrew@HURRICANE:~$ nmap 192.168.1.5 -sV --version-all
Starting Nmap 5.21 ( http://nmap.org ) at 2013-05-08 23:41 CDT
Nmap scan report for 192.168.1.5
Host is up (0.0067s latency).
Not shown: 990 closed ports
PORT      STATE SERVICE      VERSION
80/tcp    open  http         Apache httpd
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
443/tcp   open  ssl/http     Apache httpd
445/tcp   open  netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
548/tcp   open  afp?
2049/tcp  open  rpcbind
3689/tcp  open  hp-pjl      (hp-pdl probe got something back)
9900/tcp  open  upnp        TwonkyMedia UPnP (Linux 2.X.x; UPnP 1.0; pvConnect SDK 1.0)
49152/tcp open  upnp        Portable SDK for UPnP devices 1.6.6 (kernel 2.6.32.11-svn70860; UPnP 1.0)
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at http://www.insecure.org/cgi-bin/servicefp-submit.cgi :
SF-Port3689-TCP:V=5.21XI=9XD=5/BXTime=518B28FCMP=x86_64-unknown-Linux-gnu%
SF:r(GetRequest,E2,"HTTP/1.0\x20401\x20Unauthorized\r\nWWW-Authenticate:\
SF:x20Basic\x20realm=\\"forked-daapd\x20web\x20interface\\""r\nContent-Type:
SF:x20text/html;\x20charset=ISO-8859-1\r\n\r\n<html><head><title>401\x20U
SF:authorized</title></head><body>Authorization\x20required</body></html>
SF:")\r(GenericLines,108,"HTTP/1.1\x20400\x20Bad\x20Request\r\nContent-Ty
SF:pe:\x20text/html\r\nConnection:\x20close\r\nDate:\x20Thu,\x2009,\x20May\
SF:\x202013,\x2004:41:22\x20GMT\r\nContent-Length:\x20134\r\n\r\n<HTML><HEAD
SF:>\n<TITLE>400\x20Bad\x20Request</TITLE>\n</HEAD><BODY>\n<H1>Method\x20N
SF:ot\x20Implemented</H1>\nInvalid\x20method\x20in\x20request<P>\n</BODY><
SF:/HTML>\n")\r(HTTPOptions,108,"HTTP/1.1\x20400\x20Bad\x20Request\r\nCon
SF:tent-Type:\x20text/html\r\nConnection:\x20close\r\nDate:\x20Thu,\x2009\
SF:\x20May\x202013,\x2004:41:22\x20GMT\r\nContent-Length:\x20134\r\n\r\n<HTM
SF:L><HEAD>\n<TITLE>400\x20Bad\x20Request</TITLE>\n</HEAD><BODY>\n<H1>Meth
SF:od\x20Not\x20Implemented</H1>\nInvalid\x20method\x20in\x20request<P>\n<
SF:/BODY></HTML>\n")\r(RTSPRequest,108,"HTTP/1.1\x20400\x20Bad\x20Reques
SF:t\r\nContent-Type:\x20text/html\r\nConnection:\x20close\r\nDate:\x20Thu,
SF:\x2009,\x20May\x202013,\x2004:41:22\x20GMT\r\nContent-Length:\x20134\r\n\r
SF:\n<HTML><HEAD>\n<TITLE>400\x20Bad\x20Request</TITLE>\n</HEAD><BODY>\n<
SF:H1>Method\x20Not\x20Implemented</H1>\nInvalid\x20method\x20in\x20reque
SF:s<P>\n</BODY></HTML>\n")\r(Hello,108,"HTTP/1.1\x20400\x20Bad\x20Reques
SF:t\r\nContent-Type:\x20text/html\r\nConnection:\x20close\r\nDate:\x20Thu
SF:\x2009,\x20May\x202013,\x2004:41:37\x20GMT\r\nContent-Length:\x20134\r\n
SF:\r\n<HTML><HEAD>\n<TITLE>400\x20Bad\x20Request</TITLE>\n</HEAD><BODY>\n
SF:<H1>Method\x20Not\x20Implemented</H1>\nInvalid\x20method\x20in\x20reque
SF:st<P>\n</BODY></HTML>\n")\r(Hello,108,"HTTP/1.1\x20400\x20Bad\x20Reques
SF:t\r\nContent-Type:\x20text/html\r\nConnection:\x20close\r\nDate:\x20Thu
SF:\x2009,\x20May\x202013,\x2004:41:37\x20GMT\r\nContent-Length:\x20134\r\n
SF:\r\n<HTML><HEAD>\n<TITLE>400\x20Bad\x20Request</TITLE>\n</HEAD><BODY>\n
SF:<H1>Method\x20Not\x20Implemented</H1>\nInvalid\x20method\x20in\x20reque
SF:st<P>\n</BODY></HTML>\n");
Service Info: OS: Linux

Service detection performed. Please report any incorrect results at http://nmap.
```

Figure 4. Version detection

a d v e r t i s e m e n t

ICT SPRING
EUROPE 2013

BIZ STONE
Co-founder, **Twitter**

TRIP HAWKINS
Founder of **Electronic Arts**,
CEO, **Digital Chocolate**

LAURA YECIES
CEO, **SugarSync**

PETER SONDERGAARD
Senior Vice President, Research
Gartner

LUXEMBOURG

19+20 JUNE 2013

RUPERT KEELEY
CEO EMEA, **Paypal**

KOICHIRO TSUJINO
Founder Alex Corporation and developed VAIQ, Sony, former President of **Google Japan**

BRIAN STEVENS
CTO, **Redhat**

PEPE MODER
Global Director for the Digital Marketing & Communication, **Pirelli**

REGISTER NOW
WWW.ICTSPRING.COM

MORE SPEAKERS ON
WWW.ICTSPRING.COM

and the last 255 in this case is the “broadcast address”. These will be skipped by default, as hosts should not be at either address, and if they are, there is some funny stuff happening on that network. Here we go: `nmap 192.168.1.0/24 -p 80,443` (Figure 3).

We used the CIDR notation to tell nmap to scan all hosts in the network that I’m on. Then using the

```
andrewb@HURRICANE:~$ nmap 192.168.1.5 -O
TCP/IP fingerprinting (for OS scan) requires root privileges.
QUITTING!
andrewb@HURRICANE:~$ sudo nmap 192.168.1.5 -O

Starting Nmap 5.21 ( http://nmap.org ) at 2013-05-08 23:56 CDT
Nmap scan report for 192.168.1.5
Host is up (0.0028s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
548/tcp   open  afp
2049/tcp  open  nfs
3689/tcp  open  rendezvous
9000/tcp  open  cslstener
49152/tcp open  unknown
MAC Address: 00:90:A9:B1:38:44 (Western Digital)
No exact OS matches for host (If you know what OS is running on it, see http://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=5.21X0=5/8XOT=80XCT=1XCU=35361XPV=YKDS=1XDC=DKG=YKH=0090A9NTH=518
05:B2CA6XP=x86_64-unknown-linux-gnu)SEQ(SP=C7KCCD=1K1SR=CDNTI=1XCI=ZKII=1X5
05:5=5NT5=8)SEQ(SP=C8KCCD=1K1SR=CEKTI=1XCI=ZKII=1X5=5NT5=8)OPS(O1=M5B45T11
05:NM4X02=M5B45T11NM4X03=M5B4NN11NM4X04=M5B45T11NM4X05=M5B45T11NM4X06=M5B4
05:ST11)WIN(W1=16A0XW2=16A0XW3=16A0XW4=16A0XW5=16A0XW6=16A0)ECN(R=YKDF=NKT=
05:40XW=16D0XO=M5B4NN5NM4KCC=YKQ-)T1(R=YKDF=NKT=40X5-OXA=5+XF=ASXO=M5B45T11NM4XRD=0XQ-)T2
05:(R=N)T3(R=YKDF=NKT=40XW-16A0X5-OXA=5+XF=ASXO=M5B45T11NM4XRD=0XQ-)T4(R=YK
05:DF=YKT=40XW-0X5-AKA-ZNF=RKO-XRD=0XQ-)T5(R=YKDF=YKT=40XW-0X5-ZNA=5+XF=ARX
05:O=XRD=0XQ-)T6(R=YKDF=YKT=40XW-0X5-AKA-ZNF=RKO-XRD=0XQ-)T7(R=YKDF=YKT=40X
05:W=0X5-ZNA=5+XF=ARXO=XRD=0XQ-)UI(R=YKDF=NKT=40XIPL=164XUN=0XRIPL=GXRID=GX
05:RIPCK=GXRUCK=GXRU=GX)IE(R=YKDFI=NKT=40XCD=5)

Network Distance: 1 hop

OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.27 seconds
andrewb@HURRICANE:~$
```

Figure 5. Probing for the base OS

```
andrewb@HURRICANE:~$ sudo nmap 192.168.1.5 -O --osscan-guess

Starting Nmap 5.21 ( http://nmap.org ) at 2013-05-09 00:02 CDT
Nmap scan report for 192.168.1.5
Host is up (0.0018s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
548/tcp   open  afp
2049/tcp  open  nfs
3689/tcp  open  rendezvous
9000/tcp  open  cslstener
49152/tcp open  unknown
MAC Address: 00:90:A9:B1:38:44 (Western Digital)
Device type: general purpose|VoIP phone|webcam|switch|media device|HAP
Running (JUST GUESSED) : Linux 2.6.X|2.4.X (92%), Allnet embedded (90%), Allnet embedded (89%), Cisco embedded (89%), Chumby embedded (88%), Gentek embedded (88%), Siemens embedded (88%)
Aggressive OS guesses: Linux 2.6.13 - 2.6.28 (92%), Linux 2.6.17 - 2.6.31 (91%), Linux 2.4.20 (Red Hat 7.2) (91%), Linux 2.6.22 - 2.6.23 (91%), Sirio by Alice VoIP phone (90%), Linux 2.6.23 (89%), Linux 2.6.9 - 2.6.28 (89%), Allnet 2210 webcam or Cisco MDS 9216i switch (89%), Linux 2.6.19 - 2.6.31 (89%), Linux 2.6.24 - 2.6.31 (89%)
No exact OS matches for host (If you know what OS is running on it, see http://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=5.21X0=5/9XOT=80XCT=1XCU=40094XPV=YKDS=1XDC=DKG=YKH=0090A9NTH=518
05:B2DDB8XP=x86_64-unknown-linux-gnu)SEQ(SP=CEKCCD=1K1SR=CFKTI=1XCI=ZKII=1X5
05:5=5NT5=8)SEQ(SP=CDKCCD=1K1SR=CFKTI=1XCI=ZKII=1X5=5NT5=8)OPS(O1=M5B45T11
05:NM4X02=M5B45T11NM4X03=M5B4NN11NM4X04=M5B45T11NM4X05=M5B45T11NM4X06=M5B4
05:ST11)WIN(W1=16A0XW2=16A0XW3=16A0XW4=16A0XW5=16A0XW6=16A0)ECN(R=YKDF=NKT=
05:40XW=16D0XO=M5B4NN5NM4KCC=YKQ-)T1(R=YKDF=NKT=40X5-OXA=5+XF=ASXO=M5B45T11NM4XRD=0XQ-)T2
05:(R=N)T3(R=YKDF=NKT=40XW-16A0X5-OXA=5+XF=ASXO=M5B45T11NM4XRD=0XQ-)T4(R=YK
05:DF=YKT=40XW-0X5-AKA-ZNF=RKO-XRD=0XQ-)T5(R=YKDF=YKT=40XW-0X5-ZNA=5+XF=ARX
05:O=XRD=0XQ-)T6(R=YKDF=YKT=40XW-0X5-AKA-ZNF=RKO-XRD=0XQ-)T7(R=YKDF=YKT=40X
05:W=0X5-ZNA=5+XF=ARXO=XRD=0XQ-)UI(R=YKDF=NKT=40XIPL=164XUN=0XRIPL=GXRID=GX
05:RIPCK=GXRUCK=GXRU=GX)IE(R=YKDFI=NKT=40XCD=5)

Network Distance: 1 hop

OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.37 seconds
andrewb@HURRICANE:~$
```

Figure 6. Interesting results

“-p” option we listed the ports that we were interested in checking for, namely 80 and 443 the two ports commonly used for HTTP/S. You can see that nmap is quite fast at what it does, it looked across a couple hundred possible addresses in about 3.5 seconds (your mileage may vary). In that time we learned that there are at least 6 live hosts on the network and 3 of them have a web application running. When we see “open” that tells us that the port is open and an application responded. When we see “closed” that tells us that the port MAY be open, but most importantly there is not an application actually servicing the port.

Nmap can tell us quite a bit about a service and host. Next, we’ll execute nmap with service and version detection (Figure 4).

```
nmap 192.168.1.5 -sV --version-all
```

Nice! We learned what application is behind each service. Knowing that Apache, Samba are in use on this host can be quite valuable information. Using the `-sV` we enabled the service/version detection. In this case, we were running with default ports, the top 1000 so, using the `--version-all` option, will use every probe against each port that is discovered (of that top 1000).

Now let’s probe for the base OS instead of the applications. For this nmap needs to send/receive raw packets, so root privilege is needed (Figure 5).

Well, it looks like nmap couldn’t figure out what this host is running as a base OS, but notice this time it did look up the MAC address. MAC addresses are assigned to manufacturers, so based upon the MAC address we at least know what NIC is installed in the host. In this case Western Digital. Ummm Western Digital, isn’t well known for network gear, typically HDDs right?!? Let’s see if nmap can give us any more to go on about the base OS.

We will add an option to tell nmap to give us a guess (Figure 6):

```
sudo nmap 192.168.1.5 -O --osscan-guess
```

There we go. Some guesses with a percentage of certainty. Very cool indeed! First, it seems that it is some sort of appliance/device versus a server or workstation. It is likely running a Linux based embedded OS. Given the manufacturer, we can narrow this down pretty quick. It is running storage services, web services, it is a device, and the NIC is made by Western Digital that doesn’t typically make network gear. You guessed it...a WD My Book Live.

ANDREW BROOKER



If you would like to receive the custom wallpaper used for this article, you can download it for **FREE** from the EaglesBlood™ Development website.

<http://www.EaglesBlood.com>



The Bread and Butter of IT Security

Today we are going to talk about bread and butter of every IT security, networking and system professional – Nmap network scanner. Initially Nmap was a Linux command-line tool created by Gordon “Fyodor” Lyon in 1997. Nowadays it is a great set of tools with extensible framework, providing opportunity to integrate it with external scripts.

There is also a beautiful GUI called ZeNmap and editions for Windows, Mac OS X, and most UNIX OS distributions available. You can get information about all features and distributions at the official www.Nmap.org website.

Initial setup is quite straightforward. For Windows machines in most cases you just need to download the all-in-one installer, launch it as an administrator, leave all boxes checked by default and play click-click-next game.

After the setup is completed launch Nmap from the ZeNmap GUI shortcut. We will use new-school approach and show all examples in GUI. However, if you are tending to stay classic, then you can launch command prompt and navigate to Nmap.exe directory.

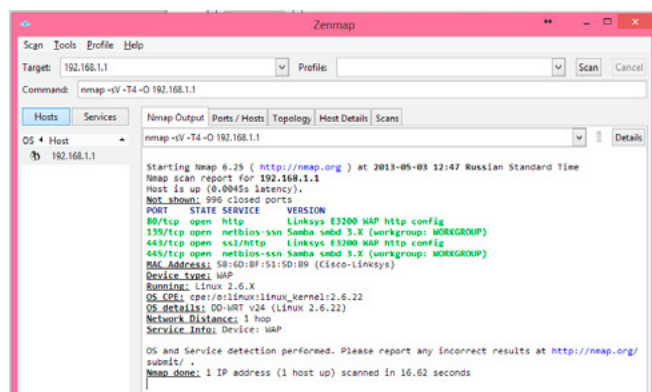


Figure 1. Scan results for my SOHO router

Your very first scan

If some Internet websites are available, then your default gateway is definitely up. Let us scan it! (Scanning localhost is not a good option as there are some peculiarities with Nmap/Windows tandem). Find out its address by typing ipconfig in command prompt and looking for default gateway value for appropriate interface. (As an alternative, you can use dummy scan target at scanme.Nmap.org). Input `Nmap -sV -T4 -O <default gateway IP>` in Command field and press Scan button. This is the output for my environment (Figure 1). Here you can see that my SOHO router:

- Is up and has some network ports open
- Is in the same network subnet, therefore network distance is 1 hop and I am able to get its MAC address
- Has a web interface available on both TCP 80 and TCP 443 ports
- Has a Samba file server included in workgroup called WORKGROUP
- Supposed to run on Linux 2.6.X kernel
- Supposed to have a Cisco/Linksys network interface based on MAC address and be E3200 router based on web interface version

How does all of this magic happen? We will provide an overview while dropping some technical details this time.

Scanning basics

Normally every device connected to a network has some network ports open and is waiting for connections. Nmap with default scanning profile tries to initiate a connection to the 1000 most used ports (Figure 2). There could be six different types of ports states

- open – actively responds to an incoming connection
- closed – actively responds to a probe but has no service running on the port, average behavior to hosts with no firewall
- filtered – typically protected by a firewall
- unfiltered – port can be accessed but no chance to determine whether open or closed
- open|filtered and closed|filtered – Nmap is tentative between two states

Please be aware that both network and security settings on target and transit infrastructure can strongly affect scan results. In this example, you can find much less details available about services. This is due to dropping `-sV` parameter, which is responsible for software vendor detection. With this parameter enabled Nmap analyzes service welcome messages, takes a “fingerprint” of the host and service behavior and compares them with the existing fingerprint database. The database can be updated at <http://insecure.org/cgi-bin/submit.cgi>. In addition, be aware that sometimes system administrators try to obfuscate against attackers. For example, this can be done by providing wrong software versions and/or product names on welcome banners. Therefore, trust no one. Especially the results of a single scan.

OS detection

Nmap is able to perform not only service’ version detection, but also OS version detection by adding the `-o` argument. This is done by a technique called TCP/IP fingerprinting which is a great achievement of the Nmap team. Nmap sends a few specially

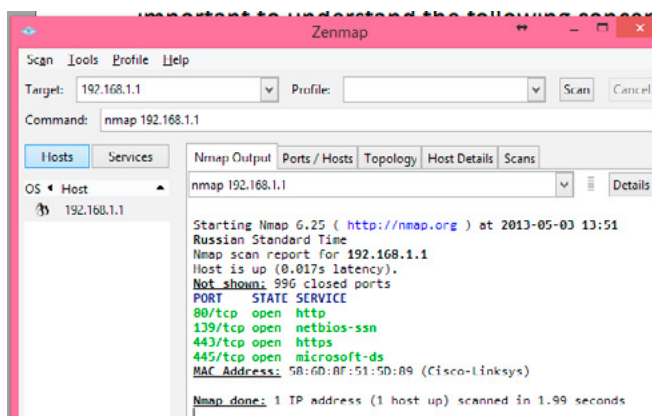


Figure 2. Scanning my SOHO router with default parameters

crafted TCP, UDP and ICMP packets to the target. On different OS versions these packets are handled in different ways. Later, Nmap analyzes the responses from the target and compares them with existing ones in the OS fingerprint database.

Staying uncovered

If you are bored enough with experiments on your default gateway, then it is time to move to others’ networks or scan your neighbors. Both of these activities are not very polite and legal, so you shall spend some efforts on staying stealthy. If you are going for more sophisticated scan types and scanning a lot of ports in a small amount of time, then there is a likely chance that you will trigger some signatures on an IDS or meet some threshold in a SIEM system. My advice is to use timing templates instead of manually tuning tons of parameters. Moreover, they are all named in a human-friendly manner:

- T0 – paranoid
- T1 – sneaky
- T2 – polite
- T3 – normal (default)
- T4 – aggressive
- T5 – insane

T0 and T1 are generally used for IDS evasion, T4 on fast channels and T5 in the occasions when you are comfortable with inaccurate scanning results. Another great idea is using the least amount of additional scan types as possible. However, if you are going to be totally impolite and lazy enough to type parameters in command-line you can simply go for `-A` parameter (aggressive), which includes `-sC`, `-sV`, `-O` and `-traceroute`. Be also aware about the existence of honeypots, which are vulnerable hosts, intentionally set up by infrastructure administrators to log all penetration attempts.

Scanning networks and groups of hosts

Network scanners are normally used by attackers to find an appropriate target and by administrators to find new and existing network hosts. Both of these tasks require scanning a significant amount of addresses. This can be done by adding the following arguments to the command-line or adding them to Target field:

- Nmap 1.1.1.1 2.2.2.2 3.3.3.3 – scan three IP addresses
- Nmap 10.1.1.1-250 – range of IP addresses
- Nmap 10.1.1.0/24 – scan subnet

You can also accomplish more complex scenarios such as taking a list of targets from a text doc-

ument, excluding some targets from the range or even scanning random targets.

Scan results can be saved for future retention, transformed by using NSE (network scripting engine) or used by some external systems like a SIEM or GRC engine. Thanks to a great GUI and the `-traceroute` parameter, we are also able to build a network overview. Here is the example of scanning the `scanme.nmap.org` host subnet (Figure 3).

Results can easily be saved by pressing the Save graphic button. Please take into consideration that by



Figure 3. Example of network map built after scanning Internet host

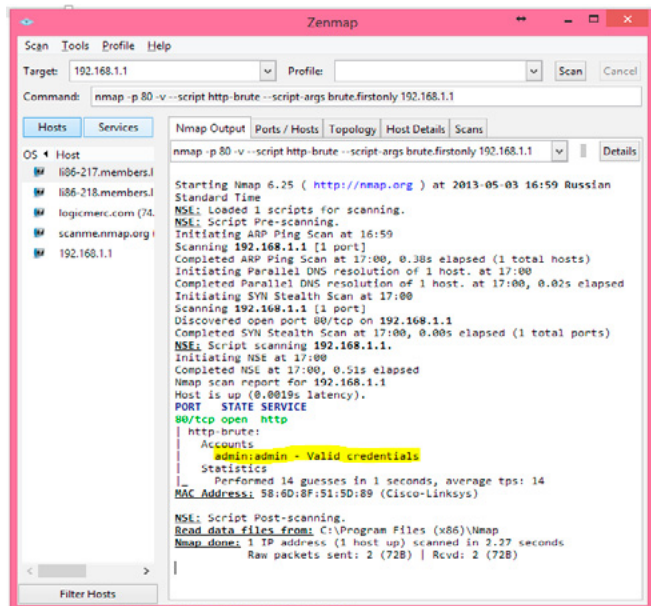


Figure 4. Output after successfully brute forcing my SOHO router web interface password

default Nmap relies on ICMP replies to check whether targets are alive. Depending on the target environment, sometimes it is better to rely on other discovery options such as IP ping, UDP ping or scanning every IP address even if there is no evidence of life.

Defining the scope of ports to be scanned

If you are not comfortable with the 1000 ports scanned by default, we can easily limit the scan with the help of the following parameters:

- `-F` – scanning 100 most used ports instead of 1000
- `--top ports [number of ports]` – to scan top [number] most common ports
- `-p [number]` – scan specific ports i.e. `-p 80,443` or `-p440-450`
- `-p [name]` – i.e. `-p https`
- `-p *` – for scanning all ports in 1 to 65535 range
- `-p U:[UDP ports],T:[TCP ports]` – to scan both TCP and UDP custom ports
- `-r` – to make port scans sequential (by default Nmap scans port randomly and then sorts them in output)

Giving a try to NSE

There are numerous features available in the product such as firewall evasion techniques, source address and port spoofing, setting flag values on both IP and transport level and many more. However, it is time to give a try to NSE bruteforce scenario and leave you on your own. First, let us change credentials to access my router to childish `admin:admin`. Then let us launch nmap with the following parameters:

```
nmap -p 80 -v --script http-brute --script-args
brute.firstonly 192.168.1.1
```

Where `--script http-brute` includes NSE `http-brute` library and `--script-args brute.firstonly` makes script to stop its run after first successful attempts: Figure 4. Here we go – credentials were found out and displayed. In scenarios that are more complex, you are able to use custom login and password databases and write your own extensions in LUA language. That is all. Hope you liked this how-to article.

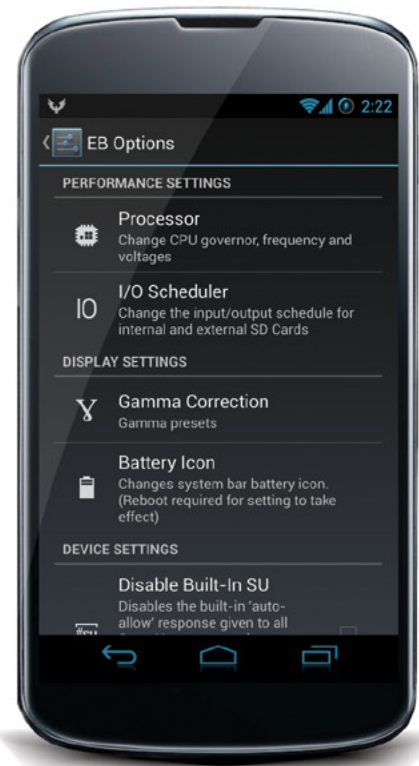
Keep tuned ;)

ANDREY MOSKVTITIN



Andrey is experienced IT security professional with 8 years of field experience and solid bunch of professional-level certificates. Currently he is employed by Microsoft and you can easily reach him via [linkedin.com/in/andreyoskvitin/](https://www.linkedin.com/in/andreyoskvitin/).

fast. stable. clean.



EaglesBlood.com

NMAP Scanning: How a Simple Tool STILL Makes Dramatic Impact

In a growing world of network analysis tools to choose from there are a few that remain just as beneficial today as they were when it first came out. NMAP definitely has held its reputation as being a go-to tool when network analyst and security researchers need it.

It's well known that if you don't at a minimum scan your network defense posture using NMAP at least once after major production changes you are taking an unnecessary gamble and risk by not doing so. While the NMAP tool hasn't significantly changed in its development lifecycle, the emphasis on using it certainly has. In this article we'll dive into the basics of doing an NMAP scan and explain some of the ways this incredible tool is able to do what it does.

According to the NMAP website (nmap.org) the scanner was designed to be used for rapidly scanning large networks. In addition to large networks, many people use it to identify security holes in single hosts such as proxy or gateway service devices. Individual usage varies depending on the need of the administrator. For example, some administrators use it to simply conduct network inventories and to identify unauthorized hosts where others may use it to identify what services are being offered by particular hosts. Additionally, it can be used for utilitarian uses such as determining how long a system has been up. There are numerous uses for this simple tool. Hollywood has even used it to beef up their Sci-fi mystique in movies like "Die Hard 4", "The Matrix Reloaded", "The Bourne Ultimatum", and nine more box office hits. It even won several awards one of which named it "The Information Security Product of the Year" by Linux Journal, Info

World and Codetalker Digest. It's a tool that is powerful, simple, and one that's not easily forgotten.

Amongst all the numerous advanced features of the tool probably the most highly utilized is the port scanning feature. A close second would be the feature used to map networks with various obstacles in the way such as: IP Filters, Firewalls, and routers. The Service and Operating System identification features would rank high in the list of uses as well. As useful as this tool is to define what a particular network is or looks like, the ultimate goal of an administrator is to use the tool to ensure these details are not available to others. Meaning that the tool is almost better used in reverse of its intended design and that the intent is to make sure NMAP can't discover much of anything at all. It's a strange paradigm but one that is readily accepted by it's developers at Insecure.org.

So how does it work? What's the best way to use it and what should you strive to achieve by using the tool? The answer is quite simply "it depends." For the sake of this article however let's discuss how the tool is supposed to work and how most people use it. First and foremost you should understand that NMAP was designed for use on an IP network. That's good because the vast majority of the networking done today employs an IP Network. NMAP uses mostly TCP and UDP protocols to gather its intelligence but also relies on things like ICMP and others as well. Since the majority of people use it for

TCP port scanning let's start there. From a network defense perspective system administrators and network administrators alike always want to know how their network perimeter appears to the outside world. What ports are open? Can we gather intelligence on what service or software controls those ports? How about version numbers? Knowing what kind of service is on the other end of those ports and the version it's running can be like gold to a hacker. Armed with that kind of intelligence a hacker can then seek out particular vulnerabilities tied to particular software and then mount an attack using that information. NMAP is a great tool to determine if this kind of information is available or not. A simple TCP SYN Stealth Scan of a host using NMAP can help identify how much and what kind of information leakage you have on your network. Let's start small and just determine what services (ports) are open and available on a server out there. Issuing the command `nmap -sS <target>` would start the NMAP scanning engine on a rampage scanning 1000 TCP ports on the target. Essentially NMAP is initiating TCP SYN connection packets to 1000 well known ports in rapid succession directed at the server and then gathering up any replies that come in the form of TCP ACK packets from that server. It does this scan in a stealthy way since it never completes what's called "The Three-Way Handshake." Typically most logging of client connections is done after full connection to the device. It's true, however, that any good Intrusion Detection System should still log the fact that one IP address sent several connections over a small span of time to multiple ports. The three-way handshake consists of a client making a connection request (TCP SYN), the server responding to the request if the port is open with a response (TCP SYN ACK),

and then finally the client issuing a response to the acknowledgement (TCP ACK). Since NMAP never sends the final acknowledgement and thereby does not complete the three-way handshake the connection isn't established and generally, therefore, not logged by the service. Here's a few screenshots of both the Zenmap tool (GUI for NMAP) and the packets NMAP fires off viewed in the Wireshark protocol analyzer (Figure 1).

The packets highlighted in purple at the top is a default ICMP echo request that NMAP does to see if the host is up and responding. Shortly after you'll notice black highlights where the client (172.20.5.101) was initiating TCP SYN packets to the server (74.220.217.163). You can quickly see that NMAP is scanning commonly used ports such as: 80, 443, 22, 445 etc. Let's take a look at where NMAP found an open port and then its response to that finding. Observe packet numbers 73152, 73161, and 73163. In packet 73152 we see NMAP sending the TCP SYN packet to the server target at port 443. Next we see the server responding with the TCP SYN ACK in packet number 73161. Finally we see NMAP sending a TCP RST in packet 73163. TCP RST is basically a reset flag used to tear down a connection quickly. In this case 443 was observed as being open and NMAP was never logged in the service running port 443 since a connection was never fully established. Now let's look at packet 73170, and 73197. Here NMAP requested a connection to the server on port 113 but the server immediately rejected the connection with a TCP RST ACK. The

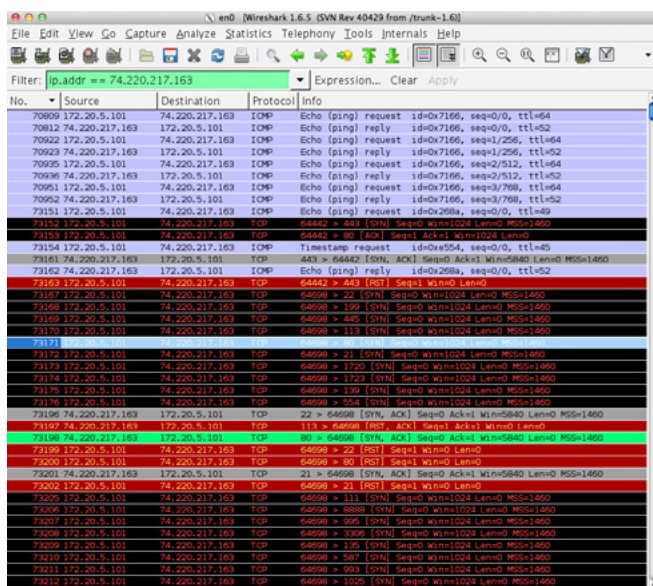


Figure 1. Wireshark showing uncompleted 3 Way Handshake

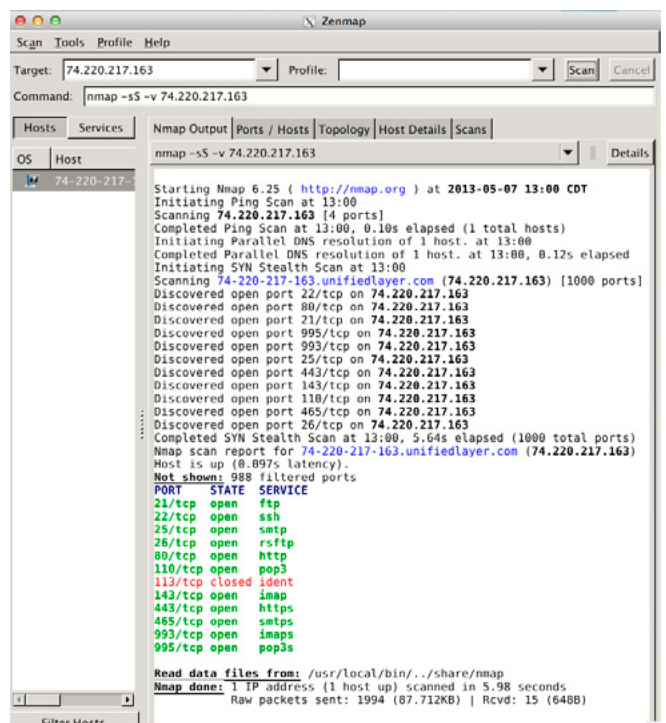


Figure 2. NMAP scan results after stealth scanning a host

server is actively telling the client this port is closed. Scrolling through the rest of the packets you can see situations where several SYN connection attempts were never responded to at all. For whatever reason port 113 was setup by the server (or filtering device) to be reported as closed where the other ports simply have no rule or service running and therefore no response generated; these are commonly referred to as filtered ports. Looking through the Zenmap results you can see what information was gathered and which ports are open on this server: Figure 2.

The NMAP SYN Stealth scan can produce one of three responses fairly reliably: Open, Closed, and Filtered. A SYN-ACK response reports that the port is open and listening. While a RST response is considered a non listener (closed). If no response is ever seen after multiple transmission attempts the port is marked filtered.

Are any of the results viewed above indicative of something bad? No not necessarily. Here we have a server that has several services running and should respond accordingly. That's great, so what would something bad look like? Let's say for example you installed MySQL on this same server recently. MySQL is a database engine and generally is installed to accompany a webserver as a back-end data store. Usually you would only want your database accessible locally from the webserver and not the outside world. We still use ports to access the database, however it's only local adapter ports that are open and therefore only local processes on the server should be able to access them. In this example if you were to run an NMAP scan of this server after that MySQL installation and found port 3306

open (default MySQL port) this would be very bad. Unless you had a specific reason for having that port open and available to the outside world it shouldn't be open and available to the outside world! In all the cases above the ports that are open and available to remote clients are designed to be. While some may argue that having some of these ports open is questionable they are still reasonable by today's standards and acceptable risks if the necessary security penetration tests have been completed.

While it's outside the scope of this article to go into the numerous other TCP scans NMAP offers it is important to highlight the differences between most of them and those differences are reported in the Table 1 (derived from nmap.org website).

Basic scanning of well-known ports is good but sometimes it's equally important to identify ports outside well-known ranges as well. After all hackers aren't going to just scan your well-known ports, they'll be looking for masqueraded ports as well. Masqueraded ports are those mapped by filtering devices to provide some security-through-obscurity techniques. For example, let's say you have an ssh server that you need available to outside clients but you don't want it available on the standard port 22 port. Instead you may have your SSH server running on port 22 on the inside of your firewall but responding to port 2222 on the outside, which is what the public sees, in an attempt to make your ssh server less of a target. For this reason NMAP allows you to define which ports to scan. Using the `-p` option you can specify what ports or port ranges you want to scan. So by issuing the command `nmap -sS -p U:53,T:1-25,80,443` your actually instructing NMAP to scan UDP ports 53 and TCP

Table 1. NMAP scan types

NMAP Scan Type	Difference from TCP SYN STEALTH
TCP Connect Scan (<code>-sT</code>)	Doesn't use raw packets, instead uses host OS to do full connection requests. No stealth since full connections are being made. Also less efficient.
UDP Scan (<code>-sU</code>)	Does UDP scanning vs TCP. Slower. If an ICMP port unreachable error (type 3, code 3) is returned, the port is <code>closed</code> . Other ICMP unreachable errors (type 3, codes 1, 2, 9, 10, or 13) mark the port as <code>filtered</code> . Occasionally, a service will respond with a UDP packet, proving that it is <code>open</code> .
SCTP INIT scan (<code>-sY</code>)	Combines most characteristics of TCP and UDP, and also adding new features like multi-homing and multi-streaming. It is mostly being used for SS7/SIGTRAN related services.
TCP Null, FIN, XMAS (<code>-sN, -sF, -sX</code>)	Exploit a subtle loophole in the TCP RFC to differentiate between <code>open</code> and <code>closed</code> ports. More info available on the nmap.org website.
TCP ACK Scan	This scan is different than the others discussed so far in that it never determines open (or even <code>open filtered</code>) ports. It is used to map out firewall rulesets, determining whether they are stateful or not and which ports are filtered.
TCP Window Scan (<code>-sW</code>)	Window scan is exactly the same as ACK scan except that it exploits an implementation detail of certain systems to differentiate open ports from closed ones, rather than always printing <code>unfiltered</code> when a RST is returned.
Zombie Scan (<code>-sI</code> <code><zombie host></code>)	AKA TCP Idle Scan. Truly blind TCP port scan of the target (meaning no packets are sent to the target from your real IP address). Instead, a unique side-channel attack exploits predictable IP fragmentation ID sequence generation on the zombie host to glean information.

ports 1 through 25, 80 and 443. As you can see the port selection is very easily defined through the command. You can also do things like `-F` (fast), `-r` (for don't randomize) and more.

As previously discussed NMAP can be used to meet a vast variety of goals. While a port scan is beneficial it's also limited on how much information can be obtained. Knowing that a webserver and smtp server is running is great but knowing exactly which version of server is running is even better. This information is integral to vulnerability analysts who are searching for footholds in an organizations perimeter. For this reason the Service and Version Detection feature of NMAP is vital to understand. Let's reverse engineer a typical Service Version Discovery scan now. Using Zenmap and the following command `nmap -sV -v <target>` we were able to discover the following information as shown in the Figure 3. As you can clearly see we have strong indicators of exactly what type of services are running on this machine as well as their version numbers in some cases. How does

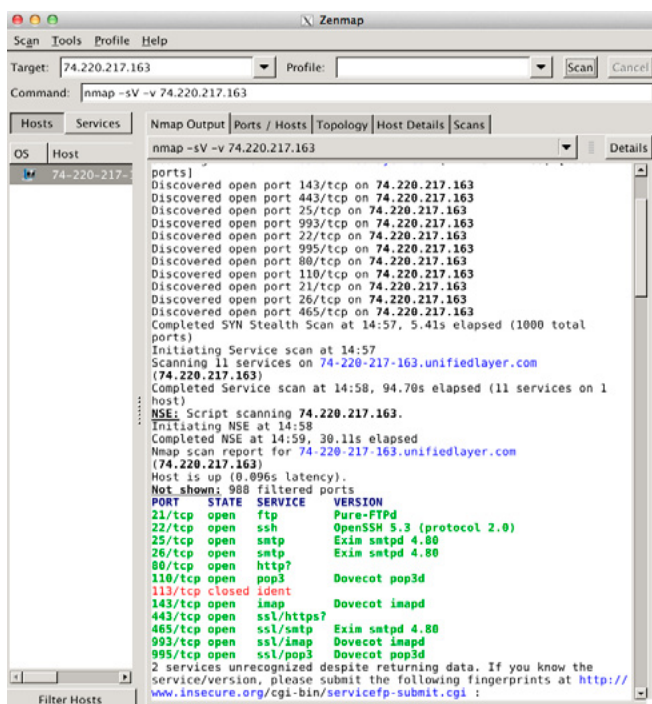


Figure 3. NMAP scan results showing Services and Version Information

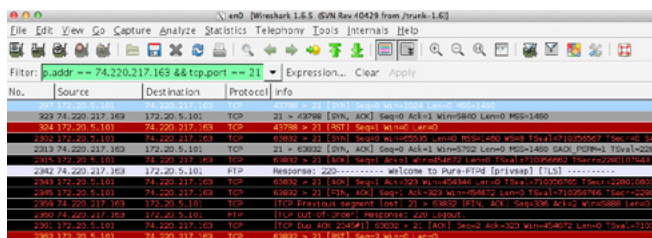


Figure 4. Wireshark showing FTP banner information in packet data

NMAP do this? Let's use port 21 as an example, which was shown to be Pure-FTPd (Figure 4).

In this case it's pretty obvious what gave away our FTP server information just by looking at the info portion of the packet information. In packet 2342 just after the three-way handshake has been established the server returns a generic information banner in the FTP protocol as response code 220. NMAP maintains a large database of signatures that reflects these patterns of common services and when it finds a match like the one observed above it's able to determine the service name and version. Now let's look at port 110 the POP Mail server (Figure 5).

Again you can clearly see in packet number 2340 that our services banner gave away the service name. You're probably saying to yourself "Really? That's it?". Most of the time this is all it takes to gain critical information that people can use against you. The obvious mitigation that needs to take place here is to remove the banner information. Any good ser-

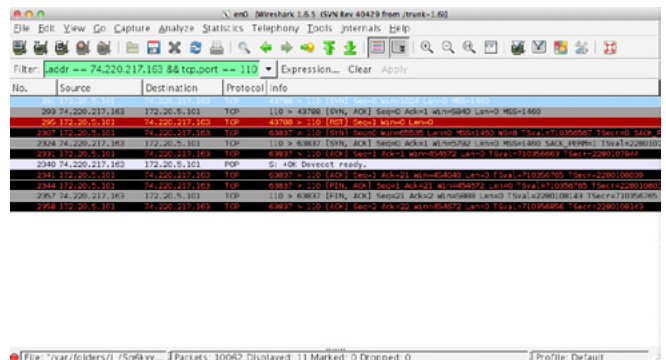


Figure 5. Wireshark showing SMTP banner information in packet data

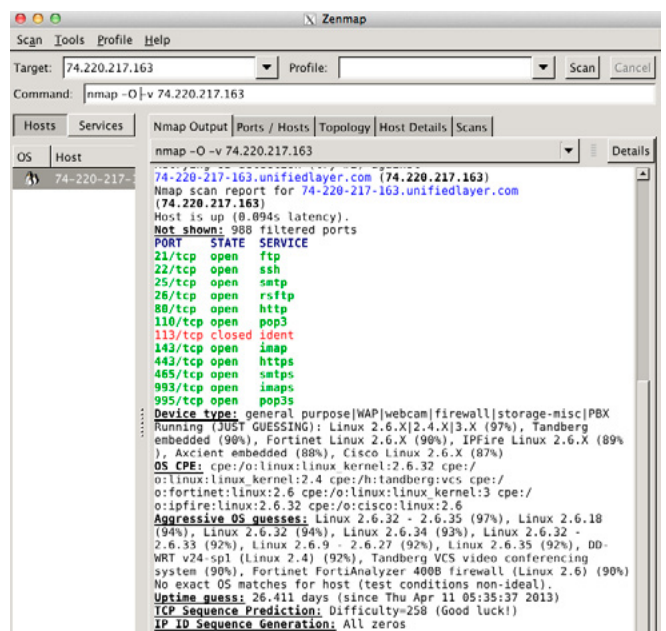


Figure 6. NMAP scan results showing O/S Discovery information of a host

vice has a configurable option to do just that. While it's not the only way people can fingerprint what kind of service is running, turning off banner information is a quick and easy way to make things much harder. Why give away information that's not necessary for public consumption anyway?

The final feature that NMAP is commonly used for is Operating System detection. NMAP OS fingerprinting works by sending several TCP, UDP and ICMP probes to known open and known closed ports of the target machine. These probes are specially designed to exploit various ambiguities in the standard protocol RFCs to help determine what type of OS is running. Dozens of attributes are analyzed in packet responses and combined to generate a fingerprint. Again the detail behind all of these tests is outside the scope of this article. There are some unordinary flags and situations that can be easily identified during a Discovery scan to help you determine if someone is scanning your hosts for information. Realize these scans are

only going to happen against a host that NMAP found as alive and up. Here's the output reported from NMAP regarding our OS Discovery scan after issuing the `nmap -O <target>` command: Figure 6.

Let's review the packets sent out by NMAP after issuing the command: Figure 7. Starting at packet number 2267 you'll see two ICMP Echo request packets issued. The first ICMP Echo has some unique characteristics. It has the DF flag turned on (Don't Fragment), TOS 0x00, Code 9, Sequence number of 295, and 120 bytes of 0x00 data with it (Figure 8-10). The second echo request has TOS 0x04, Code 0, Sequence number 296, and 150 bytes of 0x00 data (Figure 11 and Figure 12).

NMAP has sent these packets out to analyze what the response will be in the form of an ECHO Response. Linux, Unix, and Windows will respond to these types of packets differently based on the different implementation of the ICMP protocol they have employed. They're dead giveaways as to what type of OS is running behind this IP address.

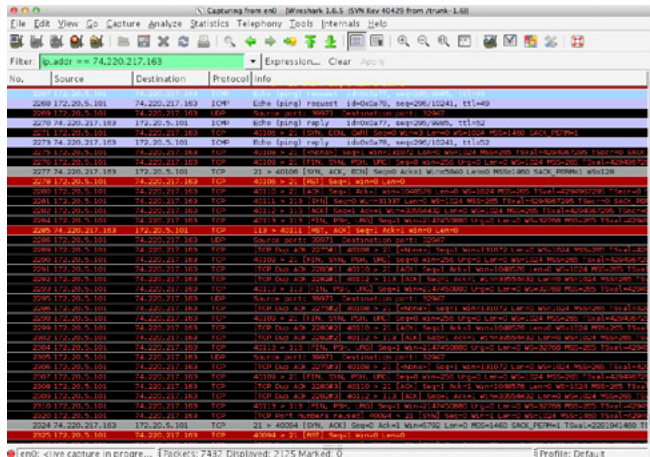


Figure 7. Wireshark showing packets initiated during NMAP OS Discovery Scan

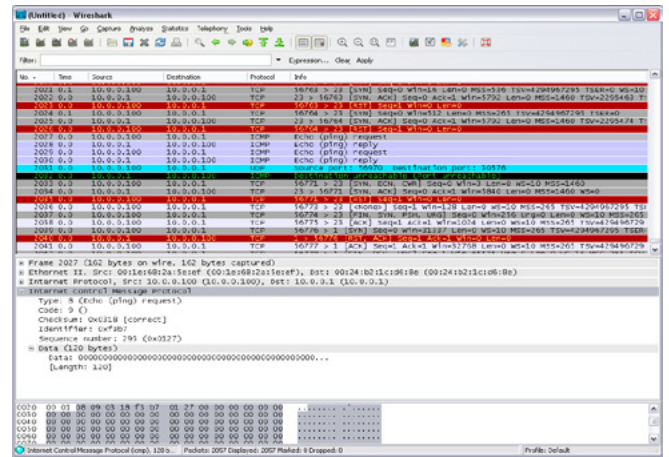


Figure 9. Wireshark Decode Window showing 2nd ICMP Echo packet details

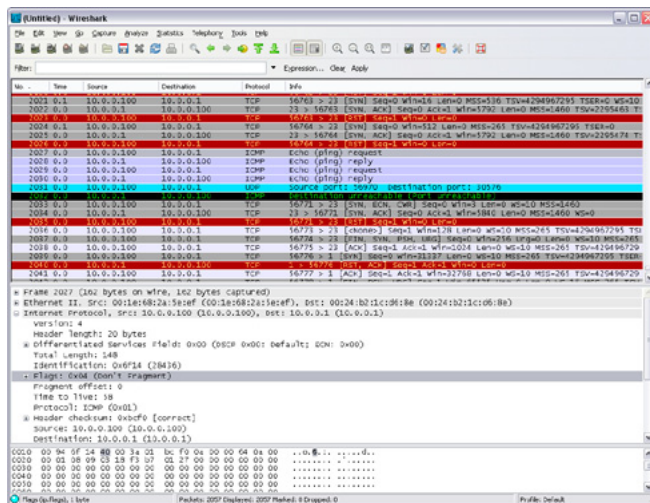


Figure 8. Wireshark Decode Window showing 1st ICMP Echo packet details

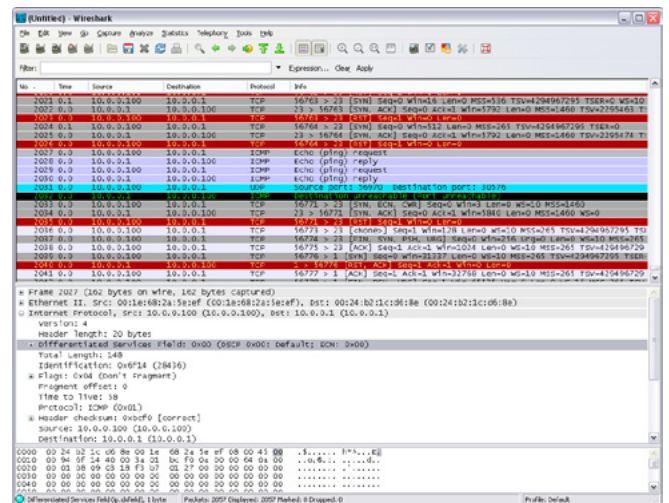


Figure 10. Wireshark Decode Window showing repeated C character (0x43)

NMAP Scanning: How a Simple Tool STILL Makes Dramatic Impact

It doesn't stop there though. In packet 2295 you'll witness a UDP packet being sent to an assumed closed UDP port with ASCII C (OX43) repeated 300 times in the payload (Figure 13 and Figure 14).

NMAP assumes that if the port is truly closed and no firewall is in between the client and the server then an ICMP port unreachable returns. You'll also note that in packet 2277 the ECN flag is set in the packet. ECN stands for Explicit Congestion Notification. In this case NMAP is utilizing this flag to test for a response to a request being made to signal network congestion problems. Default ECN handling can vary for different versions of the various operating systems that might be observed, allowing the observer to discriminate between these. Finally look at packet number 2276 where you'll see the FIN, SYN, PSH, and URG flags set. This is called the XMAS tree flags since all the lights are lit up. In XMAS tree attacks Windows machines respond with a RST packet where Unix/Linux flavored machines just ignore it. There's so much more including a series of six TCP probes all designed to do the same thing, which is to confirm a fingerprint of the

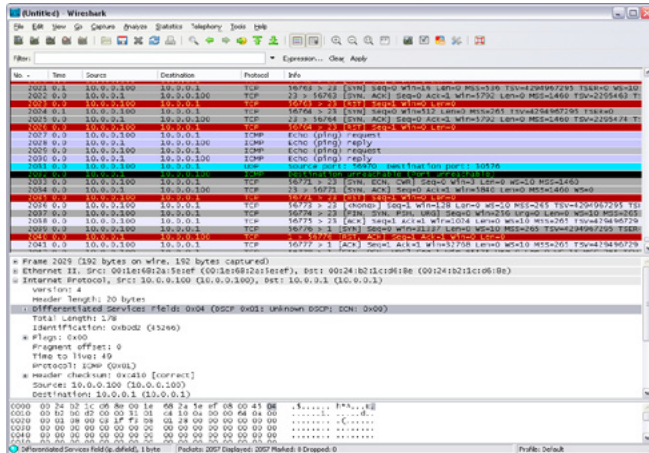


Figure 11. Second Echo Request (1)

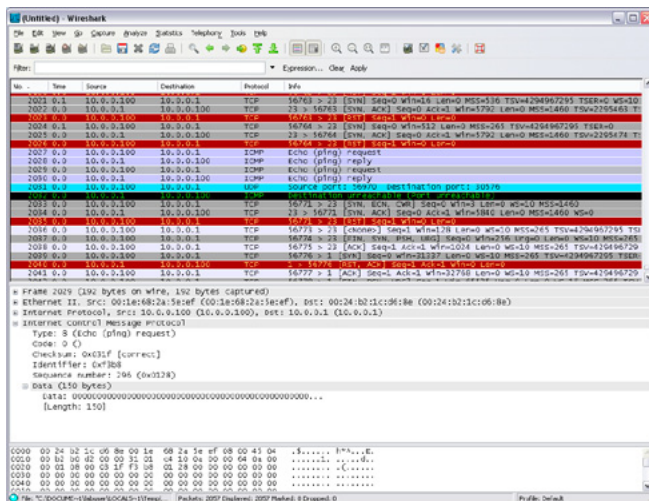


Figure 12. Second Echo Request (2)

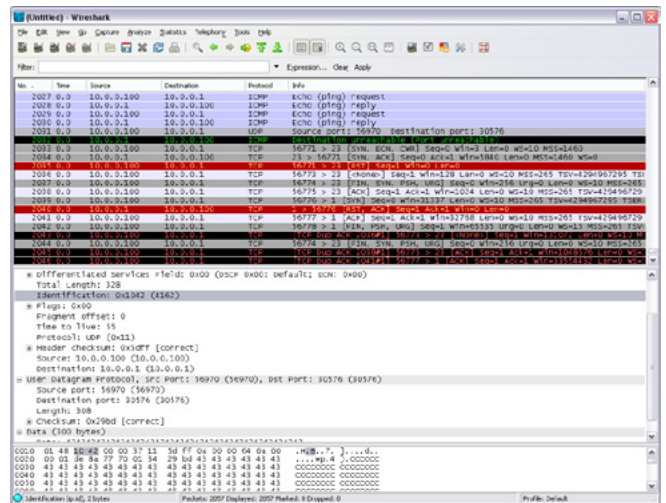


Figure 13. UDP port results (1)

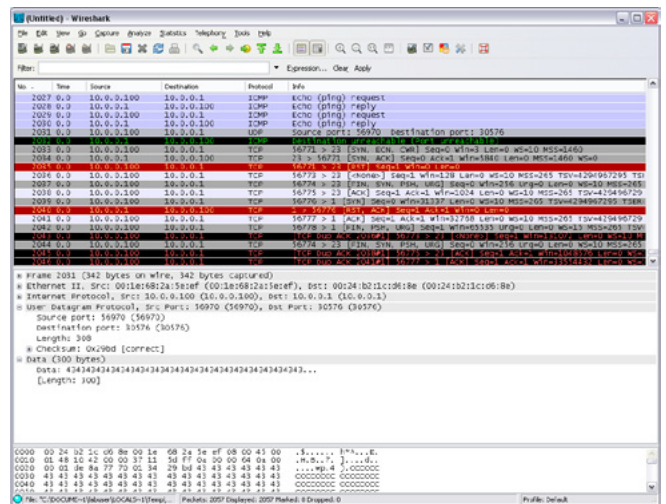


Figure 14. UDP port results (2)

Operating System by how it behaves in its responses. Aside from rewriting your operating system network stack much of this information leakage is just unavoidable. Anomalies in Operating Systems and network devices are common and will continue to be. Most security professionals just document the fact that these unique tests are being directed against them and record the suspicious behavior in case they need to do event correlation later.

NMAP is an unbelievably powerful tool. It's stood the test of time in validity and utility and probably will continue to do so for many more years to come. If you'd like to learn more about this tool you really should purchase the NMAP book that's available from nmap.org and play with it using a good protocol analyzer like Wireshark. I hope you enjoyed this brief introduction to the tool and wish you happy NMAP scanning!

NATHAN SWAIM
President, ANRC

Nmap: a “Hacker Tool” for Security Professionals

The notion of the “ethical hacker” has always been an ironic one. The developing trends of ethical hacking and offensive security have transformed the information security industry into one of the most self-perpetuating industries in the world.

The software and tools that are used to secure vulnerable information assets are the same tools that can be used to exploit them. But perhaps it's the other way around. Perhaps the tools that were created for the sole purpose of exploiting information assets are now being used to safeguard them. I suppose this is a debate that could go on forever and is really just another instance of “what came first...the chicken or the egg?”

The purpose of this essay is not to discuss the philosophical question of intent. My intentions are more modest. I merely seek to justify the importance of a tool that has been consistently labeled as malicious hacking software. The tool that I am referring to is Network Mapper, or nmap for short. Whether you are a crazed rogue agent that is bent on inciting global revolution or a network security professional (hopefully the latter, rather than the former), nmap should have a permanent place in your toolkit. Perhaps, instead of hurling criticism at an extremely functional networking tool, we should hold individuals accountable for their actions and reflect upon the well-known Benjamin Parker caveat...“with great power, comes great responsibility.” Despite some claims to the contrary, nmap is not malicious software. And I think the only reason that it is often labeled as such is because of its very impressive list of capabilities. Despite its potential to do harm, nmap can certainly play an

important role in securing a network infrastructure within a professional environment.

Nmap has steadily evolved over the years from a simple scanning utility into a full blown penetration testing platform. It can be used in every step of the security auditing process, to include network discovery, port scanning, service enumeration, vulnerability mapping and even exploitation. Throughout this article, I will discuss the capabilities of nmap as they pertain to each step in the penetration testing process.

Installation and Preparation

Obviously, prior to using nmap, it is important to have a functional version installed on the system that you are using. By default, nmap is already installed on most penetration testing and network security operating systems such as Kali-Linux, Backtrack, DEFT, Node-Zero, Security Onion and NST. However, it can also be loaded to nearly any platform of your choice. Nmap can easily be installed on all commonly used operating systems to include Windows, Linux and OSX. There is also documentation available for installing it on more obscure platforms to include BSD, Solaris, AIX and AmigaOS. Installation packages and instructions on how to perform the install on any of these systems can be found at <http://nmap.org/book/install.html>.

Zenmap – Graphical Interface

In addition to the traditional command-line interface for nmap, there is also a graphical front-end interface called Zenmap. Zenmap is also integrated into many of the different penetration testing platforms previously discussed. However, it is not installed by default in Kali-Linux (the platform that I will be using for this tutorial). Fortunately, it is in the installation repository and can easily be installed with a single command:

```
apt-get install zenmap
```

Zenmap’s point-and-click interface not only effectively streamlines what would otherwise require complicated commands but it can also be an extremely useful tool for learning how to use nmap. It uses ‘profiles’ to save commonly used scan configurations for later modification and/or use. Take a look at the image of the Zenmap interface that is provided in Figure 1 and we will briefly address each of the components.

The first component is the Target field (Figure 1. A). This field is where you can specify the remote systems that you want to run a scan against. You can enter a single IP address (e.g. 192.168.1.1), you can enter a sequential range of IP addresses (e.g. 192.168.1.0-255) or you can use CIDR notation to specify a desired subnet (e.g. 192.168.1.0/24). The second component is the Profile field (Figure 1. B). You can click on the drop down arrow to the right of this field to see several pre-configured profiles for scanning. Profiles allow you to save commonly used scan configurations for future use. You can also create profiles of your own and they will then appear in this list (we’ll discuss how to do this later in this article). The third component is the Command field (Figure 1. C). This field will indicate the command that is going to be sent to the backend. To launch the command appearing in this field, you simply click the Scan button. You can modify the value in this field directly; however, it will be automatically populated based on the values of the Target field and the selected profile. If you are new to nmap, you should pay close attention to this field, as it will provide you with a better understanding of the appropriate syntax and use of different nmap commands. The fourth component is the Host/Service list (Figure 1. D). Once you have performed scans, information will be populated in this list. If you have the Hosts button selected, all discovered host IP addresses will be listed. And if you have the services button selected, all discovered network services will be listed. The final component (Figure 1. E) is where you can sort through all of the information that has

been produced by all the scans that you have performed. The Nmap Output tab will display the exact output that would be provided if the command had been entered from the command-line interface. The Ports/Hosts tab content will vary depending on if you have the Hosts or Services button selected for your list. If the Hosts button is selected, then the information under the Ports/Hosts tab will reflect the services that were identified on the actively highlighted IP address in the list. Otherwise, if the Services button is selected, the information under the Ports/Hosts tab will identify all hosts that were found to have the particular service that is highlighted in the list. The topology tab will provide a graphical representation of the logical topology of the network, to include all hosts that have been discovered by scanning. It will use returned TTL (Time-To-Live) values to display the logical orientation of hosts, relative to one another. The Host Details tab is where you will find the bulk of information that has been discovered about a selected host. This information includes MAC addresses, IP addresses, open ports, identified services, operating system information and any additional information that has been collected. Finally, the scans tab will provide a history of scans that have been performed during the session.

At the top of the screen you will see several different drop-down menus to include Scan, Tools, Profile and Help. The Help menu is self-explanatory. The Scan menu contains options to create new scans, save scans or open previously saved scans. The Tools menu provides some additional functions that can be used to sort through and organize information collected during your scans. And the Profile menu provides options to create new scan profiles or edit existing profiles.

We will now briefly discuss how to create a new profile and then launch a scan using that profile. To get started, select the Profile drop-down menu and then select “New Profile or Command.” This will open up a profile configuration interface that can be seen in Figure 2.

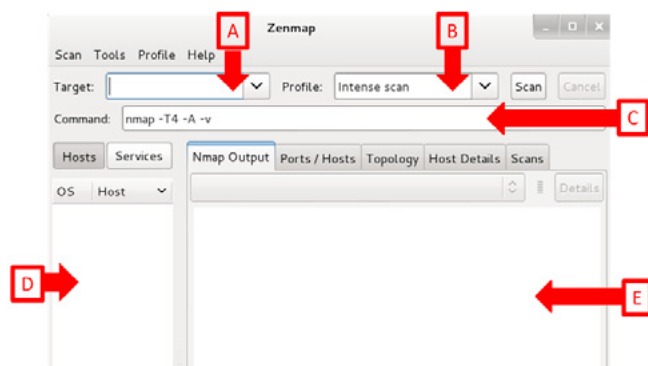


Figure 1. Zenmap Graphical Interface

The first tab (Profile tab) allows you to create a unique name and description for the profile. All the other tabs are where you will specify different configurations that will be used in your scan. As you can see in the Scan tab displayed in Figure 2, creating a scan profile is as simple as checking the boxes for options that you want to enable, entering values so that those options can be effectively employed, or selecting from pre-defined choices in a drop-down menu. If you are unsure about the function or appropriate use of any option here, you can hover over it with your mouse and the help column on the right side of the screen will be automatically populated with information on what the specific option does and appropriate values and syntax for any requested input fields. With each minor adjustment that you make to the scan configuration, the command field at the top is adjusted accordingly. This feature makes Zenmap an extremely effective tool in learning how to use nmap commands correctly. Once you have configured the scan options to your liking, you can either select the Scan button at the top right of the screen for single use, or you can select Save Changes at the bottom right. The Save Changes button will save the profile with the name provided on the first tab configurations and can then be used immediately or at a later time by selecting it by name from the Profile drop down menu on the main Zenmap interface. Although Zenmap is very powerful, you should not allow it to become a crutch that prevents you from learning the command line functions of nmap. Although Zenmap can perform many of the same functions, it still has its limitations and is not as powerful as nmap. To use nmap effectively and to its full potential, it is important to become equally familiar with the command line interface.

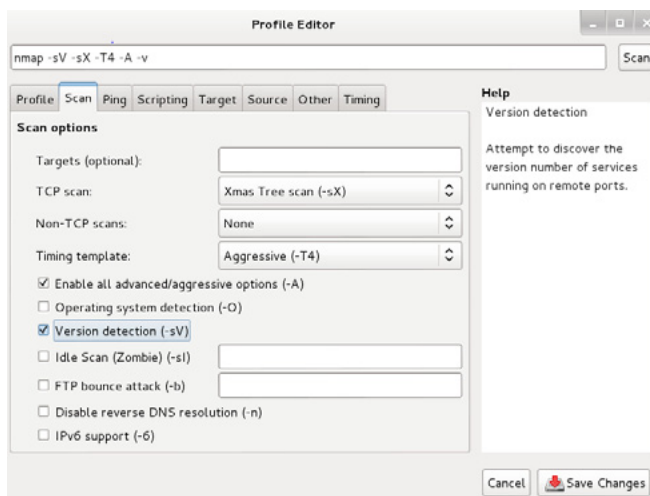


Figure 2. Zenmap Profile Editor

Network Discovery

The first step in the penetration testing process that we are going to discuss is network discovery. Prior to evaluating the security of information systems on a network, you first need to identify what you are evaluating. There are a number of different ways that you can discover hosts on a network by using nmap. In this section, we will discuss how to use nmap to perform host discovery at layers 2, 3 and 4 of the OSI model and we will also discuss the advantages and disadvantages of each. An example of a command to perform layer 2 discovery can be seen here:

```
nmap -PR -sn 192.168.1.0-255
```

The `-PR` switch specifies an ARP ping. This uses layer 2 ARP (Address Resolution Protocol) packets to identify live hosts within the specified range. It does this by sending out an ARP broadcast request for each of the IP addresses identified. If there is a live host on the network with one of those IP addresses, that host will send an ARP reply to the scanning system with its IP address and its corresponding layer 2 MAC address. Nmap will collect replies from all live hosts and then will return a list of hosts that were discovered. The `-sn` switch is used to request nmap not perform any port scanning. This prevents the transmission of any additional traffic, beyond what is necessary to perform the layer 2 host discovery sweep. Layer 2 discovery is effective because it is the quickest of all three options. However, it is limited by the fact that it can only be performed against hosts that are on the same local area network as the system that is performing the scan.

For remote hosts, you will have to use either layer 3 or layer 4 discovery. An example of a command that could be used to perform layer 3 discovery is:

```
nmap -PE -sn 109.74.11.0-255
```

This command will send a series of ICMP echo requests to each layer 3 IP address in the list. Nmap will collect all ICMP echo replies that are received and will return a list of all live hosts. This discovery scan is slower than performing an ARP ping, but it will return results for hosts on remote networks.

Finally, suppose that you are attempting to discover remote systems (so you can't use layer 2) that are behind a firewall that drops all incoming ICMP traffic (so you can't use layer 3). To address this problem, you will need to perform discovery at the transport layer (layer 4). There are several

different ways that you can use layer 4 scans to perform discovery. One effective method is using a UDP ping. An example of a UDP ping command is:

```
nmap -PU53 -sn 109.74.11.0-255
```

This command generates a series of DNS server status requests for each of the target IP addresses. A series of DNS query responses will be received from live hosts and nmap will then return these results. Because DNS is a commonly used service on UDP port 53, it is possible to identify additional live hosts by using this technique that may have been configured to not respond to ICMP traffic. Another effective layer 4 alternative to ICMP ping sweeps is to use a TCP ACK ping. An example of this can be seen below:

```
nmap -PA80 -sn 109.74.11.0-255
```

This command will send a series of unsolicited ACK replies to the specified port for all of the hosts in the IP range. Because these ACK packets were out of context of any established line of communication, live hosts will reply with a TCP RST packet to indicate that the communication should be discontinued. Nmap will collect these responses as indication of live hosts and will then return a list of them.

Port Scanning and Service Enumeration

Now that we have discovered the active IP addresses on the network that we are performing a penetration test against, we next need to identify open ports on each system and the services running on those ports. Scanning TCP ports on remote systems is the most basic function of nmap. To perform a TCP scan of a target system, use the basic command:

```
nmap 109.74.11.34
```

Like other nmap commands, this can also be used to perform scans against multiple hosts by using a sequential series (192.168.1-255) or CIDR notation (192.168.1.0/24). This standard nmap command performs a scan on 1000 commonly used TCP ports. Alternatively, you can specify a single port to scan by using the -p switch followed by the port number that you want to scan.

```
nmap 109.74.11.34 -p 21
```

This above command will scan TCP port 21 on the specified system. You can also scan a series

of ports by using the -p switch and then listing the desired ports, separating them with commas.

```
nmap 109.74.11.34 -p 80,443
```

This command will scan ports 80 and 443 on the target system. You can scan a sequential range of ports by using the -p switch followed by the first value in the range, a dash, and then the last value in the range.

```
nmap 109.74.11.34 -p 0-100
```

The above command scans the first 100 ports. To scan all 65,536 TCP ports on a target, use the following command:

```
nmap 109.74.11.34 -p 0-65535
```

You can use a TCP ACK scan to identify ports that are filtered. To do this, use the -sA switch. You can then specify a port or series of ports. No port specification will scan the standard 1,000 ports. This will then return a list of filtered ports. An example of this command can be seen below:

```
nmap 109.74.11.34 -sA
```

Penetration testers and security professionals will sometimes only scan for open TCP ports on target systems. Overlooking UDP services can cause one to completely overlook glaring vulnerabilities that might easily lead to compromise. To scan for UDP services on a target system, use the following command:

```
nmap 109.74.11.34 -sU
```

Because UDP services are not connection oriented in the same way that TCP services are, this will take longer than a typical TCP scan. The time required to complete UDP scans can be reduced by scanning for specific ports. You can specify ports the same way you had with TCP scans.

```
nmap 109.74.11.34 -sU -p 69
```

The above command performs a scan of UDP port 69, a commonly used port for TFTP (*Trivial File Transport Protocol*). It is also possible to identify the version and version number for each particular service. To do this, use the -sV switch.

```
nmap 109.74.11.34 -sV
```

This command will use a combination of banner grabbing and probe-response analysis to attempt to identify the service and version number of that service, for each scanned port. Using similar techniques, it is also possible to have nmap attempt to determine the operating system that is running on the target. To do this, use the `-O` switch.

```
nmap 109.74.11.34 -O
```

Once completed, it will either return a positive identification of the operating system or it will give a best guess and then a list of other possible operating systems.

Advanced Scanning Techniques

There is a common problem that you will frequently encounter when performing a penetration test against mid to large size enterprise networks. Most companies and organizations these days have become more security-minded and will likely have firewalls or intrusion detection systems standing between you and the systems that you are trying to scan. To scan such systems, we will need to employ some more advanced scanning techniques. One way to potentially bypass firewalls and/or intrusion detection systems is to use some traditional stealth scanning techniques that are integrated into nmap. One method that can be employed is to use a slower timing template. Timing templates range from `-T0` (paranoid scan) all the way up to `-T5` (insane scan). The lower the value of the timing template, the slower the scan will be performed. Slower scans are less likely to be flagged by intrusion detection systems. An example of a paranoid scan would be:

```
nmap 109.74.11.0-255 -T0
```

Another technique that can be used to mask your scan is to flood the network with additional decoy traffic using the `-D` switch. This function will allow you to specify multiple addresses to spoof traffic from or you can use the `RND` option to spoof traffic from random addresses.

```
nmap 109.74.11.0-255 -D RND:5
```

In addition to performing the traditional TCP port scan of the target systems, the command above will also spoof traffic from 5 random addresses to obfuscate the actual scan traffic. Another technique that can be used is to vary the packet length of your transmitted requests. Some intrusion detection systems and firewalls will drop packets based on signature packet lengths. To prevent this, you can specify the data length by using the `--data-length` switch followed by the packet size in number of bytes.

```
nmap 109.74.11.0-255 --data-length 15
```

This command will send all scanning traffic in packets that are 15 bytes in length. While this might be helpful to avoid some signature based intrusion detection systems, consistently sending packets of an unusual specified packet length could flag an anomaly based intrusion detection systems. Network intrusion detection systems will also flag traffic if you begin to sequentially connect to systems within a specified network range. To avoid performing your scans in sequence, you can use the `--randomize-hosts` switch.

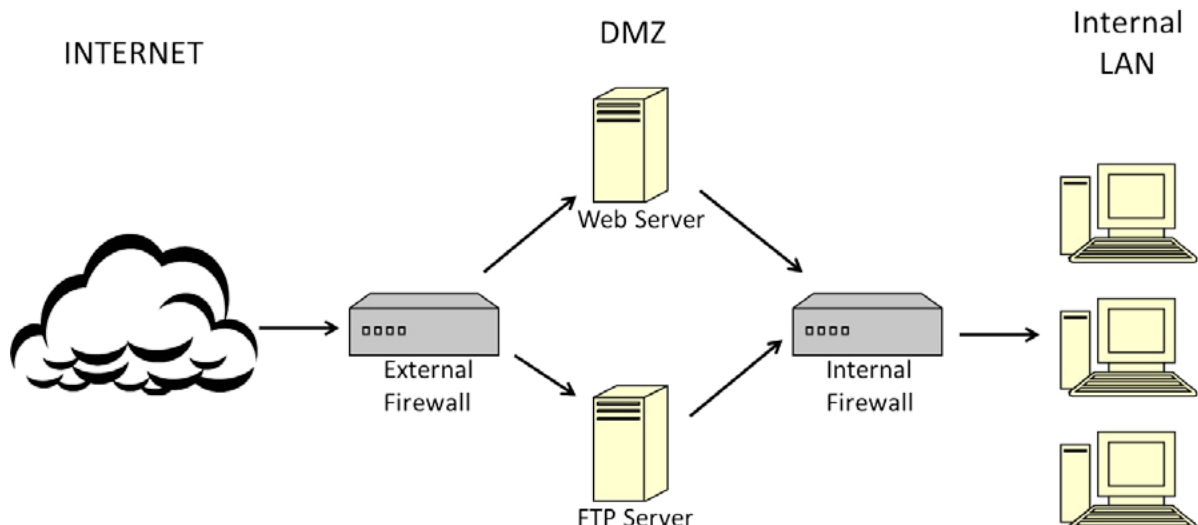


Figure 3. Common Network Configuration

```
nmap 109.74.11.0-255 --randomize-hosts
```

This command above will randomly scan each host in the range instead of performing them in sequence (109.74.11.0, then 109.74.11.1, then 109.74.11.2, etc...). While you can't spoof your IP address (because you wouldn't receive the replies necessary to determine open ports), you can spoof your MAC address. To do this, you can use the `--spoof-mac` switch, followed by a vendor ID, a specific MAC address, or 0.

```
nmap 109.74.11.0-255 --spoof-mac 0
```

Using 0 as your argument will apply a random MAC address to your scan traffic. One final technique that is worth mentioning is using the `-f` switch to fragment packets.

```
nmap 109.74.11.0-255 -f
```

Fragmenting packets will separate the data payloads of your scan traffic into multiple packets, allowing it to more easily bypass content inspection intrusion detection systems or firewalls. While all of these techniques can be effective for scanning publically accessible servers behind a firewall, they

```
Host is up (0.00090s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
MAC Address: 00:0C:29:76:12:47 (VMware)

Host script results:
|_ipidseq: Incremental!
```

Figure 4. Zombie host located with IPID Sequence Script

are usually not sufficient for scanning hosts on an internal network. While certain publically accessible servers will be available for scanning, much of the internal infrastructure will be deeper within the network and not so easily accessible. Consider the diagram in Figure 3. This is a simple example of a common configuration in enterprise networks.

The problem that we encounter here is that the internal network lies behind an internal firewall, which has stricter rules about ingress traffic, compared to the external firewall. This internal firewall is likely blocking inbound traffic from remote addresses on the web; however, it is possible that the systems in the DMZ can communicate with the internal systems. In order to be able to scan these internal systems, we must make the systems in the DMZ work on our behalf. We will discuss several different ways to do this, to include proxy scanning, zombie scanning and FTP bounce scanning.

Probably the easiest way to scan the internal systems is by using a proxy chain. Let us suppose that we have acquired access to a proxy service within the DMZ by locating an open-access proxy service, brute forcing a proxy service with hydra or by installing a proxy service on an already compromised machine. Once you have configured your system to route traffic through that proxy, you can perform nmap scans by using a full connection scan (`-sT`).

```
nmap 10.1.1.0-255 -sT
```

If there is no proxy service available, there are some clever ways that you can leverage machines within the DMZ to get scan results on internal systems. One way to do this is to use zombie scanning, also referred to as idle scanning. In order to perform a zombie scan against the systems on the internal LAN, we have to find a via-

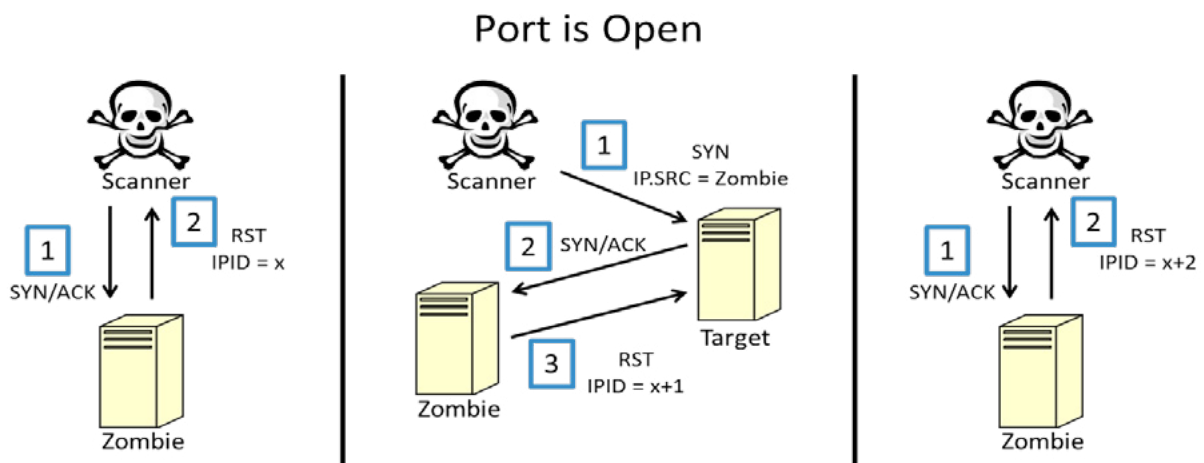


Figure 5. Zombie scan against open port

ble zombie host within the DMZ. A zombie host is any relatively idle system that uses incremental IPID sequencing. In order to locate a zombie host, we can use an nmap script to scan the DMZ for a system that fits this description.

```
nmap --script ipidseq 109.74.11.0-255
```

This script will send a series of packets to each host and track the IPID numbers for all replies received. It will then analyze these IPID numbers to classify each system as either random sequence, all zeros, or incremental. See Figure 4 for an example of the output for an incremental system.

Once we have acquired an IP address for our zombie system, we are ready to attempt our zombie scan. Prior to addressing how to perform the scan, I will briefly discuss how the scan works. Refer to Figure 5 for a diagram of what takes place when a zombie scan is performed against an open port.

First, our nmap scanner determines the current IPID value of the zombie system by sending an unsolicited SYN/ACK packet. Because no SYN packet was originally sent by the zombie system to establish a connection, the zombie then replies to our scanner with an RST packet. Our scanning system will then use the IPID of the RST packet as a reference point. Then, our scanning system will immediately follow this up with a spoofed SYN packet sent to the target system using a source IP address of the zombie system. Because the source IP address is one within the DMZ, this connection request is more likely to be able to pass through the firewall. If the target receives the packet and the destination port is open, the target will then return a SYN/ACK packet to the zombie system (who it thinks sent the original SYN packet). Because this SYN/ACK packet is received out of context, the zombie system will

then reply to the target system with a RST packet, thereby incrementing its IPID value by one. Finally, our scanner will send one last SYN/ACK packet to the zombie system. The subsequent RST reply from the zombie will increment the IPID one more time. So if the targeted port on the remote system is open, the final IPID value returned will be two numbers higher than the original value. Alternatively, Figure 6 illustrates what takes place when the port of the target system is closed.

If the port on the target system is closed, the zombie system receives no unsolicited response from the target and is therefore not instigated to send an RST packet to the target. So if the IPID value of the final RST response has only incremented by one, we can deduce that the port is closed because no SYN/ACK reply was sent to the zombie system to instigate a RST response. Otherwise, if the final IPID value has incremented by two from the original value, then we can deduce that the port on the target system must be opened because it must have replied to the zombie which instigated the response that incremented the IPID value of the zombie system.

While this may sound extremely complicated; do not become intimidated by the description, as nmap does nearly all the work for you. To actually perform a zombie scan, you just have to enter a simple command.

```
nmap -sI 192.168.199.132 -Pn 192.168.199.130
```

The `-sI` switch tells nmap to perform a zombie scan using the following IP address as the zombie host (192.168.199.132). The `-Pn` switch prevents nmap from performing an initial ICMP ping on the target system prior to attempting the scan. And the final IP address in the command identifies the

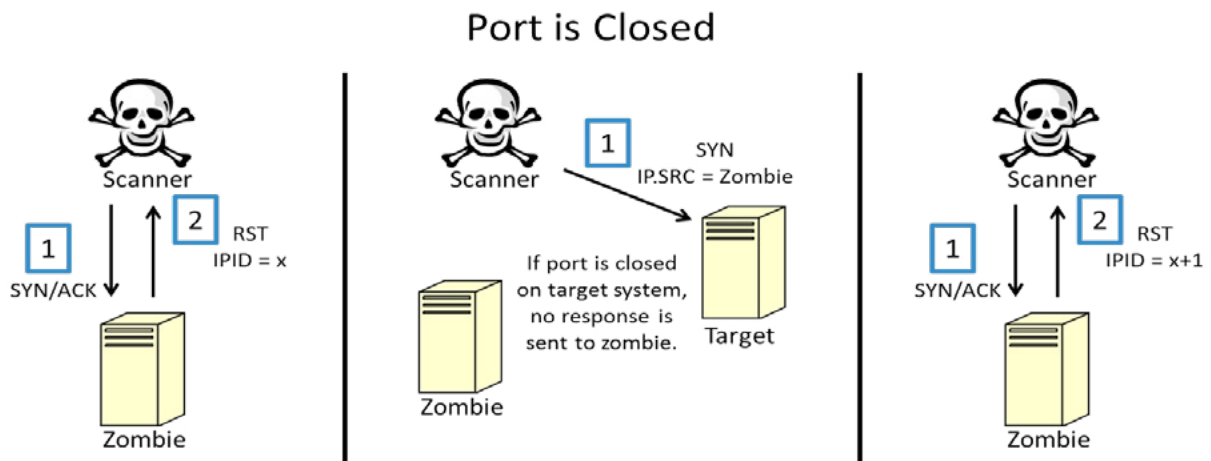


Figure 6. Zombie scan against closed port

target system. The output for this command can be seen in Figure 7.

It should be noted that the IP range that I used to demonstrate this zombie scan is a private range on my internal network. For this to work effectively against a remote network, as described in the original scenario, the systems in both the DMZ and the internal network must be on publically routable IP ranges. If the internal network is configured on a private range behind a NAT (*Network Address Translation*) server, then the nmap scanner will not be able to send the spoofed SYN packet to the internal address from its remote location.

Another way to use systems within the DMZ to attempt to scan hosts on the internal network is to use FTP bounce. Some legacy FTP servers support the capability of transmitting files to a third party system. If you have discovered an FTP server in the DMZ that allows anonymous login or that you have brute forced with hydra, you can test the FTP server to determine if it supports FTP bounce.

```

Idle scan using zombie 192.168.199.132 (192.168.199.132:80); Class: Incremental
Nmap scan report for 192.168.199.130
Host is up (0.051s latency).
Not shown: 977 closed/filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
5988/tcp  open  x11
6667/tcp  open  irc
8080/tcp  open  ajpl3
8180/tcp  open  unknown
MAC Address: 00:0C:29:C6:17:43 (VMware)
Nmap done: 1 IP address (1 host up) scanned in 34.75 seconds
    
```

Figure 7. Zombie scan against closed port

To do this, use the ftp-bounce script.

```

nmap --script ftp-bounce --script-args
ftp-bounce.username=anonymous,
ftp-bounce.password=bob@gmail.com 109.74.11.201
    
```

This command uses the nmap script with username and password arguments against the FTP server. Once completed, this scan will indicate if FTP bounce is possible or not. If the script indicates that FTP bounce is working, you can use the FTP server to perform a port scan against other systems on the network. Figure 8 illustrates how this works.

The scanner will attempt to send binary data, via the FTP bounce function, to the target system at a designated port. If the data is transmitted, the FTP server will then report back this back to the scanner, indicating that the port on the target system is open. To perform an FTP bounce scan, use the `-b` switch.

```

nmap -b ftpuser:PassW0rd@192.168.11.201:21
10.1.1.128-255
    
```

In this command, the `-b` function is used to perform a ftp bounce scan using the username "ftpuser" and the password `PassW0rd`, against the FTP server at 192.168.11.201 hosted on TCP port 21. The actions described in the diagram above will then be performed against each of the common 1000s ports on each of the target hosts from 10.1.1.128 to 10.1.1.255.

Vulnerability Mapping and Exploitation with NSE

Once you have discovered live hosts on the target network and have managed to enumerate open

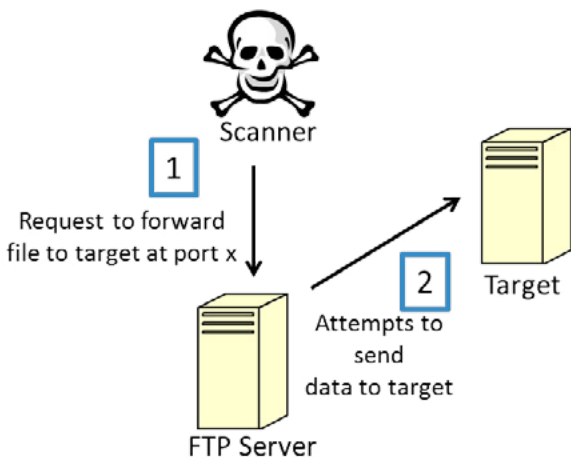
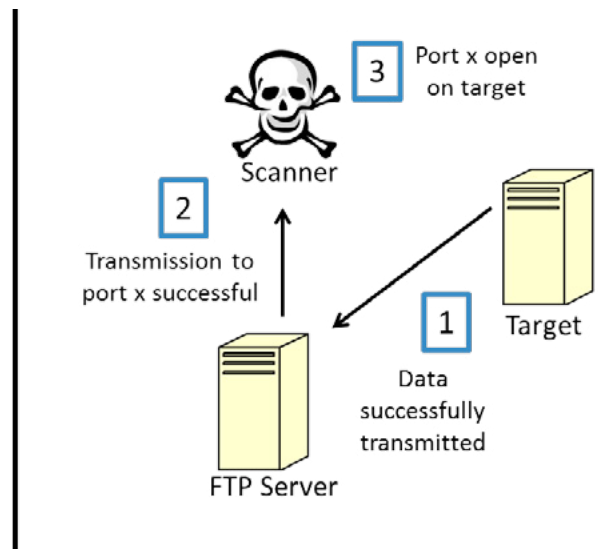


Figure 8. FTP bounce scan



ports and/or running services on those hosts, you can begin testing for and exploiting vulnerabilities. With the newly integrated NSE (Nmap Scripting Engine), there are a number of preloaded scripts that come with the standard nmap installation. These scripts have a wide range of different functions from basic information gathering (like the two that we have already discussed), vulnerability mapping, brute forcing, denial of service and even remote exploitation. All of these scripts are located in your nmap installation directory. In Kali-Linux, they can be found at `/usr/share/nmap/scripts/`. To browse to these scripts and begin working with them, use the following commands:

```
cd /usr/share/nmap/scripts/
ls
```

The `cd` command will change the directory to the location of the NSE scripts, and the `ls` command will display the contents of the directory. You can then view any of the scripts by using the `cat` command, followed by the name of the script that you want to view. For the purpose of this demonstration, we will use `ftp-vuln-cve2010-4221.nse`. To view the contents of this script, use the following command:

```
cat ftp-vuln-cve2010-4221.nse
```

Figure 9 displays the contents of this script. If you browse to the top of the script contents, you will

```
description = [[
Checks for a stack-based buffer overflow in the ProFTPD server, version
between 1.3.2rc3 and 1.3.3b. By sending a large number of TELNET_IAC escape
sequence, the proftpd process miscalculates the buffer length, and a remote
attacker will be able to corrupt the stack and execute arbitrary code within
the context of the proftpd process (CVE-2010-4221). Authentication is not
required to exploit this vulnerability.

Reference:
* http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-4221
* http://www.exploit-db.com/exploits/15449/
* http://www.metasploit.com/modules/exploit/treebsd/ftp/proftpd_telnet_iac
]]

---
-- @usage
-- nmap --script ftp-vuln-cve2010-4221 -p 21 <host>
```

Figure 9. NSE script contents

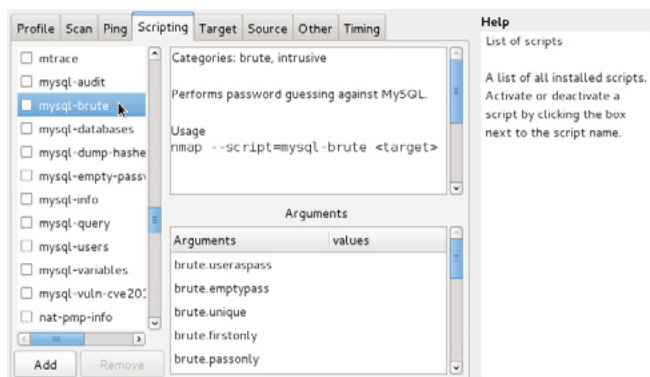


Figure 10. Zenmap Scripting Interface

see several pieces of helpful information to include a description of the script, a list of additional resources for reference, and appropriate use and syntax of the script. However, sorting through these scripts in the directory and locating a script that performs a specific function can be very tedious. In my opinion, this is where Zenmap (the graphical interface that we had previously discussed) really shines through. To demonstrate how helpful the Zenmap interface can be when working with NSE scripts, open it once again from the command line with the command:

```
zenmap
```

Then select the Profile drop-down menu and click "New Profile or Command." After the Profile Editor opens up, choose the Scripting tab at the top of the screen. Figure 10 displays an image of the Zenmap Scripting interface.

A list of all of the available nmap scripts is located on the left side of the screen. You can select any of these to have it included in your nmap command. Some scripts will require arguments in order to be able to run correctly. Post authentication scripts will often require a username and password for specific network services. In the example in Figure 10, arguments can be supplied to configure the brute force attack that is to be performed against the MySQL service. You can also easily pass the script arguments by entering the values in the Arguments window. These arguments will automatically be populated in the generated nmap command. The window in the top center provides a description of the selected script, appropriate use and syntax, and even categories to describe the functions of the script. These categories can be especially helpful if you are trying to perform a specific type of task, such as vulnerability analysis. After launching an NSE script with an nmap command, you will see the results in the standard nmap output. In the case of brute force scripts, the output will indicate if the brute force attack was successful and, if so, will display the discovered username and password. In the case of vulnerability scripts, the output will indicate if the vulnerability is present on the target system(s). And in the case of exploitation scripts, the output will provide information about the payload that was subsequently delivered to the target system.

Scripting with Nmap

In addition to its own integrated scripting engine, nmap also supports several output options that make it easy to use traditional scripting languages

for performing output analysis. Two output formats that can be useful for scripting include greppable output (`-oG`) and XML output (`-oX`). XML can be effective for higher level scripting languages that have modules that can be imported for XML parsing and greppable output can be used in conjunction with bash shell scripting to streamline analysis of nmap results. To demonstrate how this output feature could be used in conjunction with scripting, we will review a simple 4-line bash shell script, displayed in Figure 11 to analyze the results of an nmap scan.

Consider a scenario in which an nmap scan was already performed against a very large network and the output of the scan was saved in greppable format to a `networkscan.txt` file. And suppose that we want to use hydra to perform a brute force attack against all FTP services on the network, but we do not want to waste the time that would be required to scan port 21 on the entire network again. This simple script will extract all systems that have a specified port open. The first line prompts the user for a port number. The second line then assigns the value of the user input to the 'port' variable. The third line indicates to the user that all systems with that open port will be listed. And finally, the

```
#!/bin/bash
echo "Enter Port Number:"
read port
echo "Systems with port $port open"
grep $port networkscan.txt | grep open | cut -d " " -f2
```

Figure 11. Nmap Result Analysis Bash Script

```
root@Kali: ~
File Edit View Search Terminal Help
root@Kali:~# ./portenum.sh
Enter Port Number:
21
Systems with port 21 open
192.168.1.103
192.168.1.105
192.168.1.106
192.168.1.107
192.168.1.108
192.168.1.117
192.168.1.122
192.168.1.213
192.168.1.214
192.168.1.227
192.168.1.249
```

Figure 12. Nmap Result Analysis Script Output

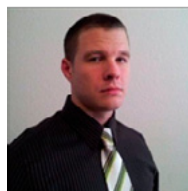
last command is where we grep out the results from the nmap greppable output. The script uses the grep command to extract all lines from the output file that reference the specified port. This will include lines that enumerate each instance of where the port is open, closed or filtered. Then, it pipes that output over to another grep function that extracts only instances in which the specified port is open. This output is then piped over to the cut function, which specifies a space character as the delimiter and then outputs the second field. By doing this, our script then outputs the IP address for each system that has the specified port open. Figure 12 displays the output of the script.

While this is a very simple script, it demonstrates how easy it can be to extract information from the greppable output format. Once acquired, this collected information could easily be used for further scripting. It could be used as a starting point to perform some other subsequent task against each system that was identified to have a specific port open.

An All-In-One Penetration Testing Tool

By combining its different capabilities, one could easily perform a complete penetration test by using nmap alone. There is no denying that nmap, with its impressive list of functions and capabilities, is a truly powerful tool. And as the years progress, it will likely become even more powerful as more people continue to contribute to this project. Like a kid with his father's gun, nmap can be a dangerous tool in the wrong hands. But if used correctly, it can be an invaluable asset for ensuring the security of your network infrastructure. So always remember to use it wisely and to use it well.

JUSTIN HUTCHENS



Justin Hutchens currently does network vulnerability analysis, intrusion detection and digital forensics for a large enterprise network with over 33,000 networked systems. He has filled numerous different roles in the Information Technology field to include network design, system development, database administration and network security. He also currently teaches courses on penetration testing with the Backtrack and Kali-Linux operating systems. He currently holds a Bachelor's degree in Information Technology and multiple professional certifications to include CISP (Certified Information System Security Professional), CEH (Certified Ethical Hacker), ECSA (EC-Council Certified Security Analyst) and CHFI (Computer Hacking Forensic Investigator).

Nmap: For Newbies

As a former Network Warfare Instructor for the US Air Force, I get asked a lot of questions: among the most common is what did you teach, or can you not talk about it? The simple answer is I taught a subset of Air Force Doctrine known as Network Defense, or NetD for short.

The premise of NetD is simple enough: the protection of information residing in, or transmitting through, network information systems (NIS). The big distinction here is that no differentiation is made between the standard TCP/IP computer networks, telephony (SS7 and cellular), radio, or even industrial control and utilities systems (ICS and SCADA). All networks are afforded equal protection, and for good reason: in today's increasingly interconnected world, these systems are converging as well. The big problem is many systems, I'll use SCADA as an example, were developed decades ago and there was no thought given to security, not because they didn't care about security, but why worry about device signing, certificate checking, and data encryption on a closed network? What does that have to do with nmap? You ask. To be blunt, everything: Nmap was one of the basic tools we would start students on. It's open source, so free, and reasonably easy to get using right away for basic network scans.

I say nmap is relatively easy to get using, but take that with a grain of salt. As you can see in the screen capture below, by running `nmap -help`, we are presented with a wealth of option flags for our use (Figure 1).

For the very new, here's the run-down on how I'm currently set-up. It's a little bit more complicat-

ed than I describe here, but these are the basics (for a complete run-down of everything, see the notes section at the end of the article). I'm running a Linux distribution called Backtrack 5 R3; Backtrack is a highly-customized Ubuntu Linux load with hundreds of tools preinstalled. These tools are designed for the professional network penetration tester (pentester) and the network security admin. However, as with all test and administration tools, they can be used for nefarious purposes. My target will be my other computer, running on the

```

root@bt:~# nmap -help
Nmap 6.01 ( http://nmap.org )
Usage: nmap [Scan Type(s)] [Options] (target specification)
TARGET SPECIFICATION:
  -Can pass hostnames: IP addresses, networks, etc.
  -Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sn: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PY[portlist]: TCP SYN/ACK/UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  --sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Mainion scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -T <scan0-5>: host(-probeports): Idle scan
  -sV/sZ: Sctp INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  
```

Figure 1. nmap has a wealth of option flags

same internal-only network, and its running Windows 7 Home Edition.

Alrighty then, lets get down to the nitty-gritty, right? Not so fast. We have to start off with the question of what is nmap? Nmap is a powerful, command-line based, open-source packet sniffer and network-mapping tool. It can determine, among a great many other things, what services and their versions are running on particular ports at a particular IP address or range of IPs, as well as what operating system and version is running. Using this tool, you can begin mapping your network and possibly identifying

rogue systems and/or services. Nmap, when determining running services doesn't simply reply on the port number, it can actually run what's called banner-grabbing to get the actual service and version number of the service.

```
root@bt:~# nmap 172.16.164.132
Starting Nmap 6.01 ( http://nmap.org ) at 2013-04-25 20:31 CDT
Nmap scan report for 172.16.164.132
Host is up (0.00032s latency).
Not shown: 995 filtered ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
2869/tcp  open  iclslap
58003/tcp open  unknown
MAC Address: 00:0C:29:B1:19:2B (VMware)

Nmap done: 1 IP address (1 host up) scanned in 17.01 seconds
root@bt:~#
```

Figure 2. Using nmap without any flags still provides some interesting results

```
root@bt:~# nmap -sV 172.16.164.132
Starting Nmap 6.01 ( http://nmap.org ) at 2013-04-25 20:32 CDT
Nmap scan report for 172.16.164.132
Host is up (0.00040s latency).
Not shown: 995 filtered ports
PORT      STATE SERVICE      VERSION
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows [un]known
2869/tcp  open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
58003/tcp open  tcpwrapped
MAC Address: 00:0C:29:B1:19:2B (VMware)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 49.04 seconds
```

Figure 3. Using this option, we can see a table of port numbers, services, and versions

```
root@bt:~# nmap -O 172.16.164.132
Starting Nmap 6.01 ( http://nmap.org ) at 2013-04-25 20:36 CDT
Nmap scan report for 172.16.164.132
Host is up (0.00043s latency).
Not shown: 995 filtered ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
2869/tcp  open  iclslap
58003/tcp open  unknown
MAC Address: 00:0C:29:B1:19:2B (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Microsoft Windows 7|Vista|2008
OS CPE: cpe:/o:microsoft:windows 7::professional cpe:/o:microsoft:windows_vista:: cpe:/o:microsoft:windows_vista::spi cpe:/o:microsoft:windows_server_2008::spi
OS details: Microsoft Windows 7 Professional, Microsoft Windows Vista SP0 or SP1, Windows Server 2008 SP1, or Windows 7, Microsoft Windows Vista SP2 or Windows Server 2008
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 26.68 seconds
```

Figure 4. Using this information, we can determine if our system is vulnerable to an attack



Figure 5. Hard rind, squishy innards (Photo credit: <http://en.wikipedia.org/wiki/File:Watermelons.jpg>)



Figure 6. In JRR Tolkien's "Lord of the Rings," he describes the city of Minas Tirith as having multiple concentric rings of defensive walls. This is layered security. As we discover in the screenplay adaption from New Line Productions though, the layers mean little when the attackers have an air force. (Photo credit: © 2003 New Line Productions, Inc. All Rights Reserved To Copyright Owner(s))

So now what do we need? Go ahead and think about it, yell out the answer when you've got it, I'll wait... Did you answer we need an IP address to scan? Is so, pat yourself on the back and go grab yourself a cookie. We need an IP to scan, and from looking at the top of the results from the `-help` option earlier, we know the nmap syntax. Let's say the IP we've chosen is 172.16.164.132, this happens to be the current IP of my Windows laptop, we type it in: `nmap 172.16.164.132` and we get our results: Figure 2.

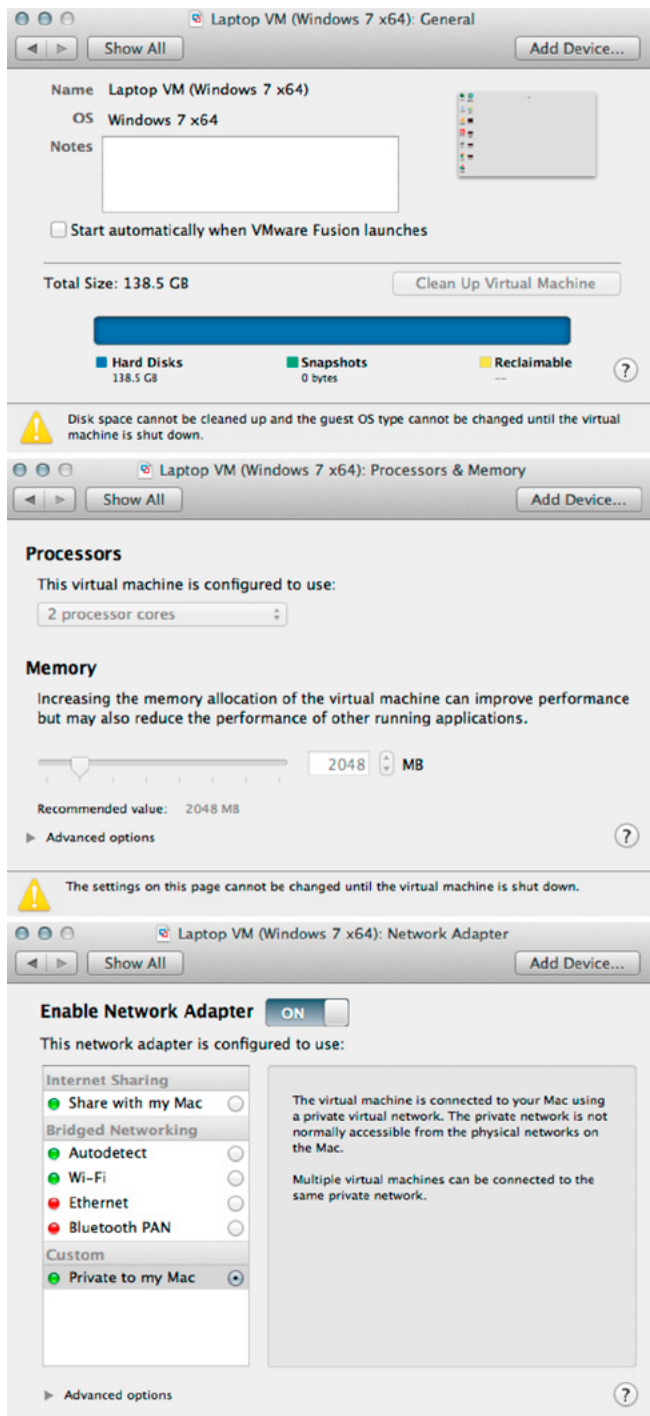


Figure 7. VM and network setup

We see now, without using any option flags, the results are pretty basic. So let's add some option flags. From the `-help` option, we know `-s` gives us the services running and `-v` gives the versions. We therefore run `nmap -sV 172.16.164.132` and we get more detailed results: Figure 3.

Some of the interesting things we see in these results are five open ports, all TCP ports: 135 is for Microsoft Remote Procedure Call [1], 139 and 445 and being used by NetBIOS [2], 2869 is running an HTTP services called Microsoft HTTPAPI [3], and 50003 is TCPWrapped [4].

All this information is a gold mine for security analysts and hackers alike. SAs use it to find the holes and unsecured systems on their networks quickly and easily. Hackers will use the information to match their exploits with potentially flawed service versions they find. Hacking, at it's most basic, can be defined as using administrative tools in ways for which they were not originally designed.

While we're on the topic, there are three types of hackers and they're divided into different colored hats: black hats are out to steal information or extort you or your business, white hats are the pentesters, these are people hired by individuals or organizations to find and exploit weaknesses in networks before the black hats get there, the third hats are the grey hats, these often blur the line by balancing a day job on the legal side with a hobby on the nefarious side.

Now let's get back to nmap. Now that I've identified the services and versions, and found there's a published exploit for the service version, I need to know if the operating system is vulnerable to the exploit. Some exploits will only work on certain services and versions running on certain operating systems and versions. From the `-help` I ran earlier, I discovered the `-o` option to get the operating system information. By running `nmap -o 172.16.164.132` I get the following results: Figure 4.

Something I will recommend doing though is running these scans on a regular basis, both from inside and outside your network (boundary firewall). If vulnerable services are found, I strongly recommend upgrading them to newer versions. If upgrading is not an option, say for legacy systems, I recommend a combination of internal firewalls and security VPNs. These will help prevent the vulnerable systems from being used as pivot points for hackers to attack the rest of your network. I've seen too many networks with very nice hard outer shells, but once inside, everything is open. I liken them to a watermelon: thick hard rind protecting the squishy innards, but get inside

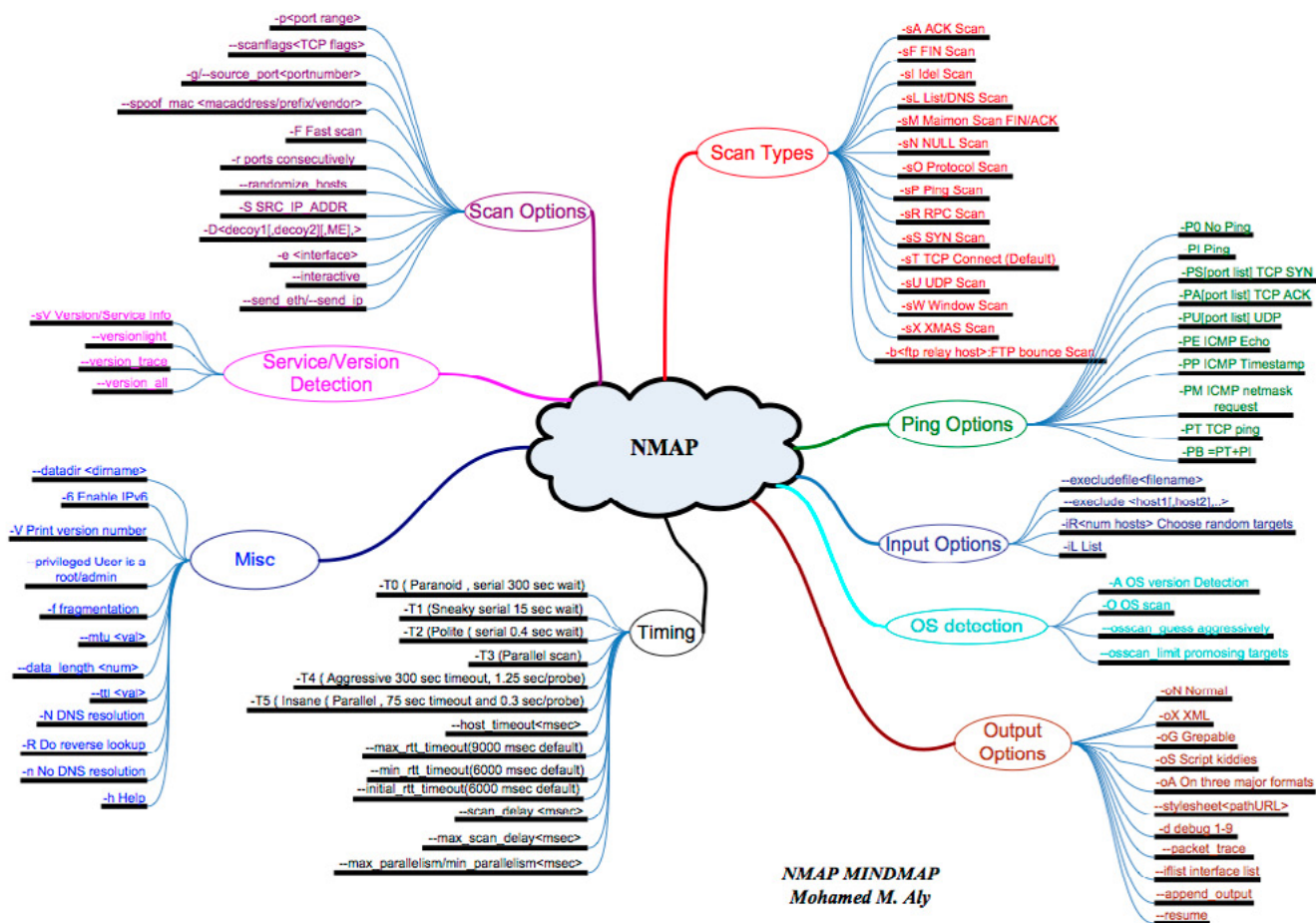


Figure 8. nmap's veritable plethora of options (Map credit: <http://nmap.org/docs/nmap-mindmap.pdf>)

and it's just that, soft and squishy. For this reason I endorse the layered security, or defense-in-depth, method of network security (Figure 5 and Figure 6).

This will conclude the beginner's guide to nmap. Look for upcoming articles on using these results to perform some white hat hacking of your own using other tools built into Backtrack: Metasploit, TFTP, John the Ripper, and others.

References

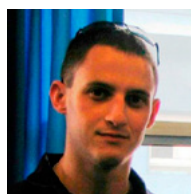
- [1] Used in programming to call sub-routines without the programmer needing to explicitly code the details of interaction.
- [2] Network Basic Input/Output System. Used to allow applications on separate computers to share information and data over a computer's local area network.
- [3] Used to allow applications to communicate over HTTP without using Microsoft's old IIS (Internet Information Server).
- [4] TCP Wrapper is a client side software solution to provide firewall features. It monitors all incoming packets to the machine and if an external node attempts to connect, the software checks to see if the node is authorized based on various specified criteria.

Notes

My set-up is actually running Backtrack 5 R3 and Windows 7 in VMware Fusion VMs on an Apple MacBook Pro 13" with 16GB of RAM. They are running on a closed virtual network with connection to the MacBook host but without connection to an outside network.

Nmap is capable of MUCH more than is described in this article. Keep playing with different options, listed in the MindMap above, to discover what else it can do (Figure 8).

ANDREW JONES



Andrew Jones (ajones@vmtraining.net) is a former US Air Force Network Warfare Defense instructor. He is now a trainer for VMTraining, specializing in cloud and virtualization technologies, as well as all types of network security.

Nmap – The Tool of Almost Endless Capabilities

Before we start out and dig in, you need to know that Nmap can be a very powerful tool in the hands of someone who knows how to use it AND has an intimate knowledge of how TCP/IP works. If you don't know some of the TCP/IP basics like IP addressing, routing, ports, and the structure of a TCP packet, it would be good idea to brush up on these skills first. As you unlock your knowledge of TCP/IP, you'll embrace the beauty of Nmap that much more.

Nmap is arguably the most well-known and widely-used information security tool in existence and the capabilities are almost endless. The original version of Nmap was written by Gordon Lyon (aka Fyodor Vaskovich or just "Fyodor") in 1997 as a simple network mapping tool, and at the time there were no future re-

leases planned. Fast forward to today; Fyodor is still writing new versions (currently version 6.25), and Nmap is no longer just a network mapping tool.

Nmap supports thousands of different features and types of scans. Among the most common features are host discovery, OS detection, version detection, firewall evasion, extensive scripting (there are now a total of 433 scripts built in; see: <http://nmap.org/nsedoc/>), and custom outputs.

"There are a lot of security people who use Nmap, but many of them don't understand its full power" – Fyodor

There are very few people who understand the full power of Nmap. We don't have the time to delve into all of Nmap's features in a single article, but I can share some basic scans and some advanced scans that we use every day in our work as professional pentesters.

At this point, we'll assume that you already have Nmap installed on your system. If not, installing it is a piece of cake; <http://nmap.org/book/install.html>. For readers who prefer a GUI, check out Zenmap; <http://nmap.org/zenmap/>.

The commands are the same whether you type them into a terminal window or the Zenmap window.

```
root@bt:~# nmap scanme.nmap.org

Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-07 17:31 EDT
Nmap scan report for scanme.nmap.org (74.207.244.221)
Host is up (1.3s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
514/tcp   filtered shell
646/tcp   filtered ldap
9929/tcp  open  nping-echo

Nmap done: 1 IP address (1 host up) scanned in 84.88 seconds
root@bt:~#
```

Figure 1. Nmap interface overview

```
root@bt:~# nmap -sS -T4 scanme.nmap.org

Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-07 17:54 EDT
Nmap scan report for scanme.nmap.org (74.207.244.221)
Host is up (1.2s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
514/tcp   filtered shell
646/tcp   filtered ldap
9929/tcp  open  nping-echo

Nmap done: 1 IP address (1 host up) scanned in 66.37 seconds
root@bt:~#
```

Figure 2. A quick scan

Basic Scans

The default Nmap Scan

Even without any options, Nmap gives us some pretty good information. Using Nmap without any tags allows Nmap choose the scan type, the timing details, the target ports, the output formats, the source ports and addresses, and much more. Nmap is almost too easy to use; `nmap -iR` lets Nmap even choose the targets! (Figure 1)

NOTE: It's important to run Nmap as a root user. We are using a test BackTrack machine in this example, logged in as root.

SYN Scan

`nmap -sS -T4 scanme.nmap.org`

A simple SYN scan (`-sS`) with the “aggressive” timing template applied (`-T4`). This output looks almost identical to running Nmap without any options applied (Figure 2). A SYN scan sends a TCP segment with the SYN flag set and waits for a response (a segment with the SYN and ACK flags set). There is no completion of the connection to the port by Nmap, meaning that Nmap doesn't send the final ACK segment to complete the three-way handshake. This is the most common type of Nmap scan. In this basic example, we didn't feed Nmap with a list of ports to scan, so Nmap scanned the “most common” 1000 ports for each protocol. These ports are based on research conducted by Fyodor and can be found in a file named `nmap-services` (default location: `/usr/local/share/nmap`) (Figure 3). The results indicate that TCP ports 22, 80, and 9929 are open, and that TCP ports 514 and 646 are “filtered”. Filtered means that something is in between Nmap and the target that prevents Nmap from confirming whether a port is open or closed. We want to know if TCP ports 514 and 646 are actually open or closed. For this we need to run a few more basic scans and combine the results.

FIN Scan

`nmap -sF -T4 scanme.nmap.org`

This scan has the FIN flag set (`-sF`) and all ports responded as `open:filtered`. The FIN scan is good to test through firewalls or packet filtering devices that don't maintain state. Stateless devices typically only filter on the first packet in a standard communication; a packet with the SYN flag set (`sS`), so a FIN scan can sometimes tell us a little more information about the system. Here we can tell that the system is most likely behind a stateful packet filtering device (Figure 4). Laying the first scan results over this scan doesn't help us. We still see filtered in our results for the ports in question; TCP 514 and 646. What happens when we send traffic with the ACK flag set?

ACK Scan

`nmap -sA -T4 scanme.nmap.org`

At first glance it looks like the same result as the FIN scan, but one major difference; all ports are reported as “unfiltered” now. If we combine our results thus far, it looks safe to say that TCP 514 and 646 are probably closed (Figure 5).

Knowing what ports are open on a host is good to know, but let's take this a little further. Let's have Nmap try to tell us what type and version of system is listening on the open ports.

If you are still interested in knowing why Nmap reported TCP 514 and 646 as filtered in the original scan, you can add the `--reason` option (Figure 6).

Scan Multiple Hosts

There are numerous ways to scan multiple hosts, including:

- `nmap 192.168.0.1,192.168.0.5,192.168.0.12` – Scans the three hosts noted.
- `nmap 192.168.0.1-254` – Scans all of the hosts in the range.
- `nmap 192.168.0.0/24` – Scans all of the hosts in subnet.
- `nmap -iL scanlist.txt` – Scans all of the hosts noted in the `scanlist.txt` file. The file can be named anything.

```
GNU nano 2.2.2 File: nmap-services
# THIS FILE IS GENERATED AUTOMATICALLY FROM A MASTER - DO NOT EDIT.
# EDIT /nmap-private-dev/nmap-services-all IN SVN INSTEAD.
# Well known service port numbers -- mode: Fundamental; --
# From the Nmap Security Scanner ( http://nmap.org )
#
# $Id: nmap-services 30210 2012-11-07 21:34:43Z fyodor $
#
# Derived from IANA data and our own research
#
# This collection of service data is (C) 1996-2011 by Insecure.Com
# LLC. It is distributed under the Nmap Open Source license as
# provided in the COPYING file of the source distribution or at
# http://nmap.org/data/COPYING . Note that this license
# requires you to license your own work under a compatible open source
# license. If you wish to embed Nmap technology into proprietary
# software, we sell alternative licenses (contact sales@insecure.com).
# Dozens of software vendors already license Nmap technology such as
# host discovery, port scanning, OS detection, and version detection.
# For more details, see http://nmap.org/book/nan-legal.html
#
# Fields in this file are: Service name, portnum/protocol, open-frequency, optional comments
tcpmux 1/tcp 0.001995 # TCP Port Service Multiplexer [rfc-1070]
tcpmux 1/udp 0.001236 # TCP Port Service Multiplexer
compressnet 2/tcp 0.000013 # Management Utility
compressnet 2/udp 0.001845 # Management Utility
```

Figure 3. Most common ports

```
root@bt:~# nmap -sF -T4 scanme.nmap.org
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-07 18:00 EDT
Nmap scan report for scanme.nmap.org (74.207.244.221)
Host is up (0.00022s latency).
All 1000 scanned ports on scanme.nmap.org (74.207.244.221) are open:filtered
Nmap done: 1 IP address (1 host up) scanned in 4.06 seconds
root@bt:~#
```

Figure 4. FIN scan

```
root@bt:~# nmap -sA -T4 scanme.nmap.org
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-07 10:07 EDT
Nmap scan report for scanme.nmap.org (74.207.244.221)
Host is up (0.00016s latency).
All 1000 scanned ports on scanme.nmap.org (74.207.244.221) are unfiltered
Nmap done: 1 IP address (1 host up) scanned in 0.18 seconds
root@bt:~#
```

Figure 5. Lets look at the results

Version Detection and OS Fingerprinting

`nmap -sS -PN -A -v -T4 scanme.nmap.org`

There is some very good information gained from this scan, but let's look at the scan options real quick first. We used a few new options in this scan that we didn't use in our prior scans; `-PN` (no ICMP ping of the host first), `-A` (enabled OS detection, version detection, script scanning, and traceroute), and `-v` (verbose, which allowed us to monitor what was going on during the scan) (Figure 7).

NOTE: If we only wanted OS and version detection, we could have used `-sV` (version detection) and `-O` (OS detection) options instead of the `-A` option.

As you can see from the results, this system is running OpenSSH version 5.3p1 and Apache version 2.2.14. We can also see that our fingerprinting was not entirely successful (note the numerous

```
root@bt:~# nmap --reason scanme.nmap.org
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-08 23:10 EDT
Nmap scan report for scanme.nmap.org (74.207.244.221)
Host is up, received reset (1.2s latency).
Not shown: 995 closed ports
Reason: 995 resets
PORT      STATE SERVICE REASON
22/tcp    open  ssh     syn-ack
80/tcp    open  http    syn-ack
514/tcp   filtered shell   no-response
646/tcp   filtered ldap    no-response
9929/tcp  open  nping-echo syn-ack

Nmap done: 1 IP address (1 host up) scanned in 31.97 seconds
root@bt:~#
```

Figure 6. Scanning results explained

```
Completed Parallel DNS resolution of 2 hosts. at 15:51, 0.06s elapsed
NSE: Script scanning 74.207.244.221.
Initiating NSE at 15:51
Completed NSE at 15:51, 2.17s elapsed
Nmap scan report for scanme.nmap.org (74.207.244.221)
Host is up (0.19s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 5.3p1 Debian 3ubuntu7 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey: 1024 8d:60:f1:7c:ca:b7:3d:0a:d6:67:54:9d:69:d9:b9:dd (DSA)
|_ 2048 79:f8:09:ac:d4:e2:32:42:10:49:d3:bd:20:82:85:ee (RSA)
80/tcp    open  http     Apache/2.2.14 ((Ubuntu))
|_ http-favicon: Unknown favicon MD5: 156515D43C0F7DC6E2493BDC43F795
|_ http-methods: GET HEAD POST OPTIONS
|_ http-title: Go ahead and ScanMe!
514/tcp   filtered shell
646/tcp   filtered ldap
9929/tcp  open  nping-echo Nping echo
Device type: general purpose/storage-nisc/VoIP phone
Running (JUST GUESSING): Linux 2.4.X (90%), Microsoft Windows 7 (96%), Bluearc embedded
OS CPE: cpe:/o:linux:linux_kernel:2.4 cpe:/o:microsoft:windows:7::enterprise cpe:/h:blue
Aggressive OS guesses: DD-URT v24 sp2 (Linux 2.4.37) (98%), Microsoft Windows 7 Enterprise (91%), Pirelli DP-10 VoIP phone (88%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
TCP Sequence Prediction: Difficulty 258 (Good luck!)
IP ID Sequence Generation: Incremental
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 7. Version detection

```
root@bt:~/usr/local/share/nmap# nmap -sT -p 80 -oG - 76.12.173.16-31 | grep open
Host: 76.12.173.21 (sns03atn.atnhost.net.br) Ports: 80/open/tcp//http//
Host: 76.12.173.23 (sup03atn.atnhost.net.br) Ports: 80/open/tcp//http//
Host: 76.12.173.26 () Ports: 80/open/tcp//http//
```

Figure 8. Finding all web servers

```
Nmap scan report for scanme.nmap.org (74.207.244.221)
Host is up (1.1s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 5.3p1 Debian 3ubuntu7 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache/2.2.14 ((Ubuntu))
514/tcp   filtered shell
646/tcp   filtered ldap
9929/tcp  open  nping-echo Nping echo
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 118.49 seconds
root@bt:~/usr/local/share/nmap#
```

Figure 9. Using a decoy

guesses). The scan took 55.45 seconds to run, which seems reasonable, but maybe you're impatient like I am. I don't really need reverse DNS resolution; disabling this (using the `-n`) option, saves me 13 seconds. This may seem inconsequential, but if you're scanning hundreds or thousands of hosts it could make a big difference.

Find All Web Servers

`nmap -sT -p 80 -oG - 76.12.173.16-31 | grep open`

New options again, and three systems responding on TCP port 80 (usually HTTP) within the IP address range given to Nmap. Connecting to each of these hosts with a browser can confirm the results. The options used in this example are (Figure 8):

- `-sT`, this is a full TCP connect scan using the three-way handshake. This is a very reliable scan.
- `-p 80`, this specifies that only TCP port 80 will be scanned.
- `-oG`, greppable output makes for easy searching.
- `- 76.12.173.16-31`, the IP address range to be scanned. Note that in the command there is a space between the G and `-` and another space between the `-` and 7.
- `| grep open`, feeds the output to grep searching for the word open.

Evasion Through Deception

`nmap -sS -sV -D adobe.com,playboy.com,yahoo.com,whitehouse.gov scanme.nmap.org`

Similar results to our previous scans of *scanme.nmap.org*, the major difference is we used *adobe.com*, *playboy.com*, *yahoo.com*, and *whitehouse.gov* as decoys. Assuming that there is an intrusion detection system in place, our real traffic would be intermingled with the decoy traffic. There's only one problem with the scan as I configured it; chances are pretty good that my source IP address would stand out easily amongst Adobe, Playboy, Yahoo, and the White House IP addresses. If you use the decoy (`-D`) option, be sure to pick decoy hosts/IP addresses that closely match yours. One important note about the decoy option; you should pick decoy hosts that are actually online or you could end up taking the target offline (Figure 9).

Nmap Scanning From a Zombie

Another favorite for evasion or hiding the true identity of our scan is idle scanning. In idle scanning, the target doesn't ever know that we scanned them because Nmap spoofs the source IP address. The target sees the traffic as originating from the spoofed IP address and not ours. The spoofed

source IP address belongs to a “zombie” host. The zombie system must be online and reachable by both the target and our scan machine. I’ll help you find a zombie a little later. First, I’ll briefly cover the idle scanning process.

Idle scanning is a three-step process.

Step 1 is determining the zombie’s IP ID; first Nmap sends a SYN/ACK packet to the zombie. The zombie is not expecting a SYN/ACK (a response to traffic the zombie never sent) and responds with a RST. In the RST packet is the zombie’s IP ID. Nmap records the IP ID.

Step 2 is sending a forged SYN packet to the target to whatever destination port that we want to scan. The source IP address is forged as the zombie’s IP address, and the target responds (or doesn’t) to the zombie. The zombie does not expect a SYN/ACK packet (the response to our SYN packet) from our target and responds to the target with a RST packet. This RST packet sent from the zombie to our target has an incremented IP ID.

Step 3 is determining the zombie’s new IP ID. Nmap sends another SYN/ACK packet to the zombie. Again, the zombie is not expecting the packet and responds with a RST. This RST packet has a new IP ID. If Nmap receives a packet with an IP ID that has increased by two, then the port is open. If the IP ID has only increased by one, then the port is closed.

Nmap.org has a great explanation of the idle scan along with graphics online (<http://nmap.org/book/idlescan.html>).

Now we need to find a good candidate for a zombie. We’re going to use a built-in Nmap script to help us.

nmap -iR 1000 —script ipidseq -T5 -v -oA zombiefile

Using these options, Nmap will scan 1000 random IP addresses using the ipidseq script. The ipidseq script will test the hosts to see if they are useable as zombies. I sped up the scan using the -T5 (insane) because I don’t want to wait longer than I have to and I’m not too concerned about being quiet right

```
root@bt:~# nmap -iR 1000 --script ipidseq -T5 -v -oA zombiefile
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-09 05:51 EDT
NSE: Loaded 1 scripts for scanning.
NSE: Script Pre-scanning.
Initiating Ping Scan at 05:51
Scanning 1000 hosts [4 ports/host]
```

Figure 10. Idle scanning

```
root@bt:~# nmap -Pn -p- -sl 63.151.132.207 scanme.nmap.org
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-09 06:19 EDT
Idle scan using zombie 63.151.132.207 (63.151.132.207:80): Class: Incremental
```

Figure 11. Hiding your identity

now. The -oA zombiefile option will output the results to three files; zombiefile.xml, zombiefile.nmap, and zombiefile.gnmap. I’ll take my pick from this output and conduct an idle scan (Figure 10).

Look for printers in your output; printers make for great zombies!

nmap -Pn -p- -sl 63.151.132.207 scanme.nmap.org

The idle scan is a simple way to hide our true identity when scanning. In this scan we are using the -sl (idle scan) option with the zombie’s IP address (in this case a home computer somewhere) to mask our scan of the scanme.nmap.org host (Figure 11).

Conclusion

As I mentioned at the beginning of this article, Nmap is a very powerful and extremely flexible tool. I showed you some of the basics and a few cool things that Nmap can do; now the rest is up to you. I encourage you to check out the Nmap Network Scanning book (written by Fyodor), the free version of the book online (<http://nmap.org/book/toc.html>) and the detailed manpage for more. The more you use Nmap, the more you will develop your own favorite scan types and options.

Just some of the things that we didn’t have the time or space to cover here include advanced firewall (and IDS) evasion, performance tuning, packet traces, the various output types, and scripting using the built-in NSE (Nmap Scripting Engine).

In closing, some words of wisdom:

“If I’ve learned one thing from writing Nmap, it’s that the networks can be tricky and unpredictable. You never really know for positive what’s going on.” – Fyodor

NOTE: Scanning systems and/or networks that don’t belong to you is rude and illegal in some places. Scan systems that you have permission to scan.

EVAN FRANCCEN



Evan Francen is a passionate information security expert who serves businesses of all sizes, in all industries by cooperatively solving the complex issues surrounding information security. He is considered to be an “information security evangelist”.

NMAP – Get To Know the Network

NMAP is a network scanner but not a security measure. The main aim of this software is to perform host and services discovery and network reconnaissance. The initial release written by Gordon Lyon also known as Fyodor Vaskovich (if you watch Defcon talks) was back in September of 1997. Fyodor keeps the NMAP project rolling which today gives us version 6.25 thanks to an active user community.

NMAP is a network scanner but not a security measure. The main aim of this software is to perform host and services discovery and network reconnaissance. The initial release written by Gordon Lyon also known as Fyodor Vaskovich (if you watch Defcon talks) was back in September of 1997. Fyodor keeps the NMAP project rolling which today gives us version 6.25 thanks to an active user community.

If you are reading this article thinking that you don't know what NMAP is and you have never seen it before there is a great possibility that you already have seen it and there is even greater possibility that people such as your parents have seen it too. The reason behind it being that NMAP has been featured in many movie hits over the years including – Matrix Reloaded, Dredd, Bourne Ultimatum (my personal favourite), Die Hard 4 and several more.

NMAP is available across many computer platforms including Windows, Linux and Mac OS X. NMAP is also available on less popular (for general public) operating systems such as FreeBSD, Sun, HP-UX, and IBM's AIX. Even the old school Amiga is capable of running NMAP if you have all the time in the world to get the source code to run on it. There are also mobile versions of NMAP working on Android and iOS platforms which can be beneficial. Why? I will explain later. Another very common en-

vironment would be academia where NMAP is often part of educational networking programmes.

Ethical issue behind NMAP

NMAP can have its uses for all types of “hats” out there – blackhat, whitehat, greyhat. As this is Hackin9 magazine I really shouldn't have to explain what the different hats mean. In the ideal world with everyone abiding by rules and ethics, an NMAP user would use the software primarily to audit the network for maintenance/monitoring and security improvement. Although this being a double edged sword we could also use it to audit the network we are trying to gain unauthorised access to. In today's world where trust within the society is rather a difficult issue, NMAP use by default is associated with bad and rogue. ISPs often discourage network port scanning and will probably quickly get in touch with you within the first 15 minutes of your scan to state your intentions – even when good they still won't like it. It is hard to find a “no questions asked” ISP these days. I would know being an Internet Service Provider myself. The main information gathered by NMAP includes:

- Status of network ports (open, closed, filtered, unfiltered, open|filtered, closed|filtered),
- List of services,
- Operating system type,

That information can be used to track inventory of the network or be used to prepare an assault on one as data extracted by NMAP can have many uses. To explain it in simple terms a blackhat would use the information to launch a network attack of probably high success, a whitehat would use it for asset discovery, security profiling, and a greyhat would use it for penetration testing.

Types of Scans

NMAP offers a range of scans within its capability starting from quick scans all the way to very intensive and heavy rather long ones. I will try to describe this relatively easy but not getting into too

much detail so you should really know your TCP/IP at this stage although not to worry if you are a beginner or in the internet world – a noob, after reading this article you should be able to gather and most importantly interpret some results and take some decisions on what to do next.

SYN Scan (-sS)

This being a default scan makes it also the most common and popular one, and has many advantages such as speed; as it is capable of scanning hundreds or even thousands of IP addresses per second. Scanning not being the most favourite activity on the network by the “network police” should be relatively stealthy and non-intrusive as no one wants you to sniff around their network without a fundamental reason for doing so. That is where SYN scan comes in handy as it never opens a full TCP connection but just gets enough in order to see if the port is listening then giving it appropriate status within the scan results.

TCP Scan (-sT)

This would be a default choice if SYN scan is not suitable for your needs. Although, if you can use SYN, then usually that would be a better option. The disadvantage is that the TCP scan opens a full connection where a SYN doesn't. So the result of both scans is the same, but TCP scan connections are more likely to be logged and the administrator or IDS of a given system / network will know that someone is snooping around.

UDP Scan (-sU)

UDP being less popular than TCP would not be scanned as often although it is still possible. If you would want to gather information about services such as – DNS, DHCP or SNMP, then this would be the suitable type of scan. UDP scans are generally slower and often missed during the network audits. Unfortunately, UDP services are a common target for attackers to exploit.

NMAP offers a few other types of scans which are mainly a combination of TCP and UDP with additional tweaks to extract different types of information about the machine or a network you are trying to scan. Fundamentally it all narrows down to port and services discovery. The described basic scans should be enough to get you going.

All types of scans within NMAP are based on que-rying IP addresses in some way, an IDS is most likely going to detect the activity of a continuous ping. To prevent IDS detection based on volume you can configure NMAP to be less aggressive while scanning by simply choosing options such: Figure 3.

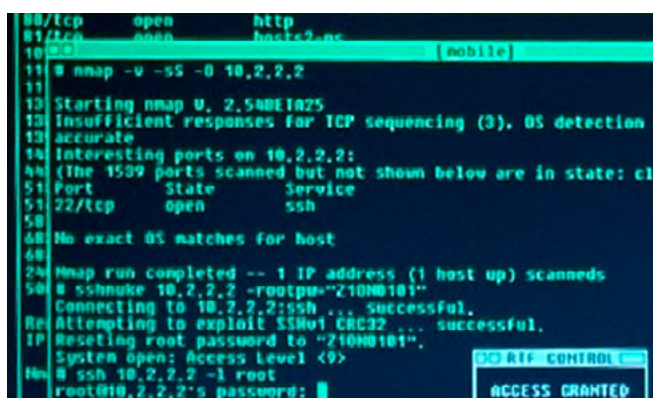


Figure 1. NMAP featuring in Matrix Reloaded

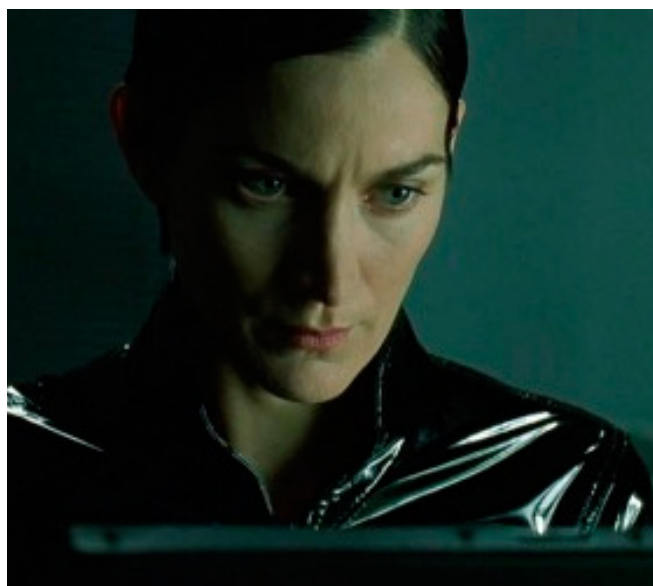


Figure 2. Matrix Reloaded

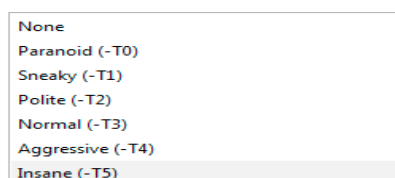


Figure 3. Types of NMAP timing

Scenario Use

Ping Sweep

Ping sweep technique would be used to simply discover echo replies from different hosts utilizing ICMP. This would be the first step to scan the network in order to determine what is plugged into it and possibly discovering what services are running. Ping sweep is most likely going to get the attention of the intrusion detection system, although thanks to NMAP you can easily spoof an IP address and by impersonating a trusted machine on the network you can simply get away with it. Some routers can block some of the ICMP traffic although we can pipe scans over e.g. port 80 for example by adding simple command option `-PT80` that should give us some information if the overall scan didn't.

Operating System Detection

Each software package on the planet has its weaknesses. If let's say a blackhat rogue user would target your machine, he wouldn't run all the possible exploits against it. The first step would be to establish the OS and services and tailor their attack specifically for that platform or configuration.

Discovering Unused IPs

NMAP has proved its use within the good side of IT too and for example can be used to see which IPs on the network are not used and can give a potential to reuse them.

Useful Options

You can log all of the scans into a text file if you like by simply appending `-oN [filename]` to the command.

Another useful argument to use is `-D` which stands for decoy which will allow you to hide and alter the IP address of the machine you are scanning from to impersonate another host.

I am also very interested to see how NMAP copes on a 3G mobile network, which I will also be testing while undertaking this research.

Tutorial

The installation of NMAP is fairly straightforward without any complicated steps. A Windows installation would just require launching the executable which can be downloaded here: <http://nmap.org/download.html>.

Linux installation can vary as users prefer different package managers. For this article I have picked the simple and easy to get Ubuntu Linux distribution where you can simply install NMAP by typing the following command within the terminal window:

```
sudo apt-get install nmap
```

In the tutorial part I will include screenshots from a Windows operating system. NMAP within a Windows environment is rather more comfortable to use and doesn't require a user to remember as many commands as they can simply be constructed within the profile editor.

As my target for scanning I will pick my own ISP to see what potentially I can get from it. I don't expect much as the users around me probably use just the internet but you never know! We will see what we can discover then see if we can channel some effort into it. And to be very honest with all of you guys I wrote this article on my laptop while sitting in the garden, relaxing in the sunshine. You might think that this won't be very professional as I am not writing it relying on a network lab full of virtual machines and so on. I am trying to show that you can adapt NMAP to apply within any Ethernet enabled situation. As I mentioned in the article before, use of NMAP on your Android or iOS device would be beneficial in environments such as coffee shops or libraries. I recently ran a scan from my iPhone while waiting in the lobby of a company that I was interviewing for. I was waiting for a good 20 minutes and none of my scans were interrupted. This clearly shows that the IT personnel were probably preoccupied with something else 😊. I guess with a small device such as a mobile phone, you can get away with it, however, I am sure it would be a different case if I sat in there with my 17" laptop workstation with a high gain dBi Wi-Fi antenna attached to its monitor. Security would possibly question what I was doing because surely it wouldn't look like you are checking your e-mail.

This tutorial will reflect some of the steps described within the scenario use (section above).

To initiate a ping sweep from nmap we have to construct a command:

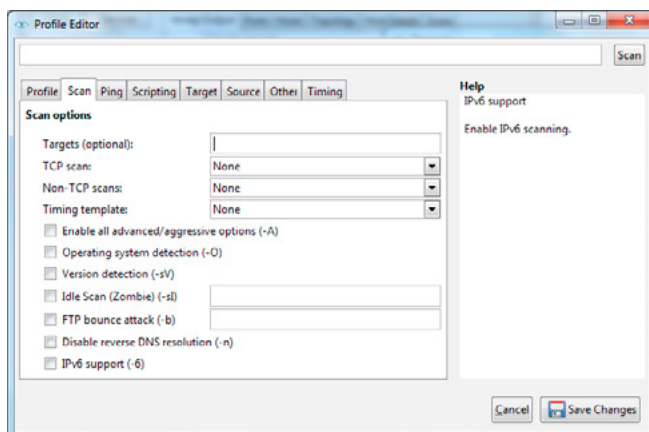


Figure 4. Profile Editor within NMAP


```
Nmap -sP ip
```

In my case I will go ahead and scan the class C subnet.

```
nmap -sP *.*.*.0/24
```

Shortly afterwards first results should start to kick in: Figure 5.

So in simple terms I can see which addresses are up and we can progress with scanning for services running on them utilizing the SYN scan technique. From the list of hosts I gathered I have picked a random IP and run the following command against (Figure 6):

```
Nmap -sS IP
```

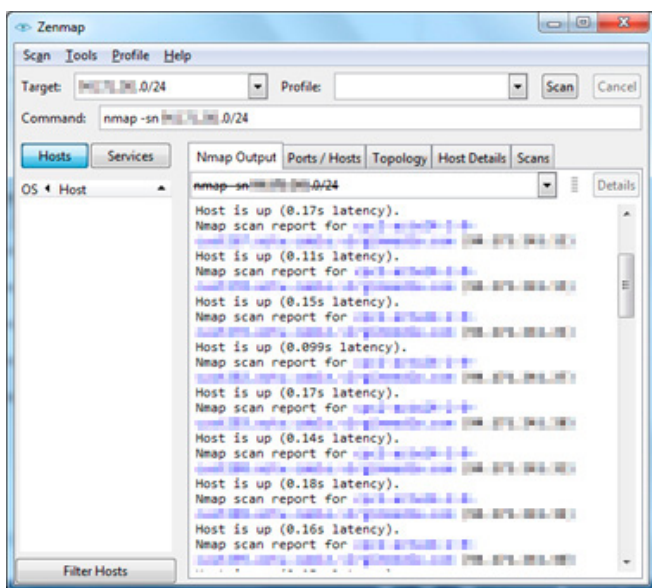


Figure 5. Ping Sweep Results

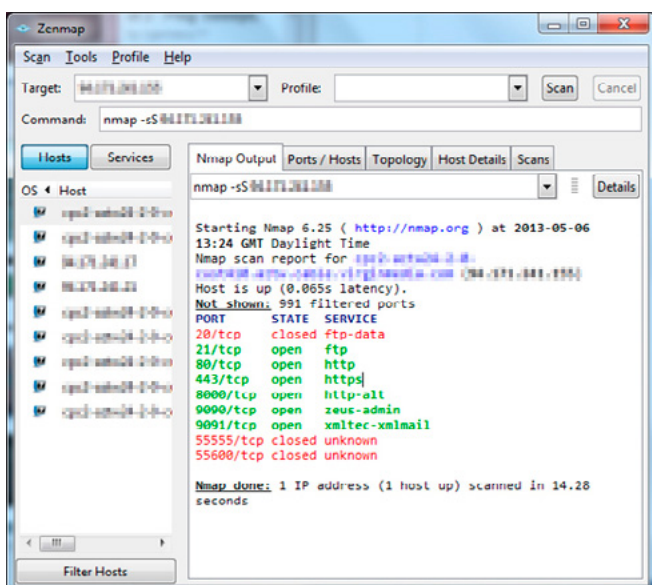


Figure 6. NMAP SYN Scan Results

From the results we can see that there are several types of ports open and most of them are HTTP ports so let's just quickly pop this IP address into the browser see if it comes up with something! And BINGO here is the result: Figure 7.

Whoever is behind this IP is using LG Network Storage appliance which is a NAS product. If I wanted to exploit this further the next step would be to try the `admin:admin` combination against the authentication mechanism but I won't!

How would this discovery go down in the "hats" world?

Blackhat – would try to download as much information about the given appliance in order to break the security and gain access to it to perhaps extract the data residing on the hard drives.

Greyhat – would simply make the discovery then be happy with their work and leave it alone (like I did).

Whitehat – would probably feel bad for this person being exposed to such a threat and would help to configure their network so the device would not broadcast this information to anyone unless someone is accessing this from outside their house which also is unlikely because you wouldn't run such an appliance on dynamic addressing.

I can also try to detect which operating system is running on the device although the IP being the gateway I would probably get some generic results. But I will attempt it anyway. The command for OS detection is very simple (Figure 8):

```
nmap -O IP
```

The 96% guess is most likely to be right running DD-WRT which is a common Linux base for routers and switches. I am sure that they are not running a HP fibre channel SAN within their home 😊.

I have connected my iPhone to the network and scanned for it and it turns out that it has open ports.

```
PORT      STATE SERVICE 62078/tcp open  iphone-sync
MAC Address: 3C:D0:F8:04:55:AC (Apple)
```

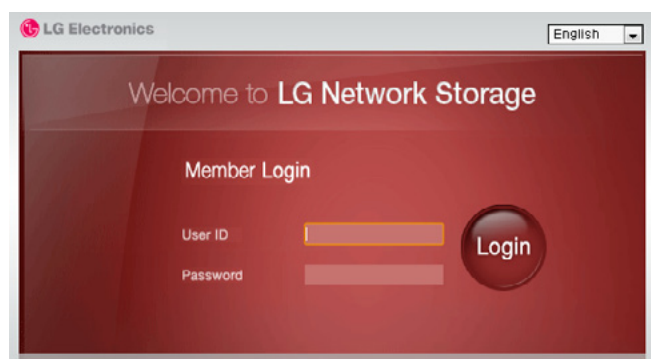


Figure 7. LG Network Storage Login Screen

I believe this one is used for sync over Wi-Fi.

I was also very interested in how NMAP would interpret the mobile phone's OS with the `-o` option:

```
Device type: media device|phone
Running: Apple iOS 4.X|5.X|6.X
OS CPE: cpe:/o:apple:iphone_os:4
cpe:/a:apple:apple_tv:4 cpe:/o:apple:iphone_os:5
cpe:/o:apple:iphone_os:6
OS details: Apple Mac OS X 10.8.0 - 10.8.2 (Mountain
Lion) or iOS 4.4.2 - 6.0.0 (Darwin 11.0.0 - 12.2.0)
```

There are no lies here as I am running one of the iOS' described by NMAP above. I wonder if we could produce a statistic based on scanning a single subnet of a 3G network to compare amount of users who have Apple and Android devices.

The next step will be to hotspot into my iPhone and scan the mobile network to see what NMAP can do.

So at this stage my laptop is connected to my iPhone and I have the public IP address record for that too. I will paste it into NMAP and run a ping sweep and a SYN scan against the items found.

Simple ping sweep found 2 hosts that are up within 91.55 seconds. Now it is time to see what these hosts are running and check if perhaps they might be mobile devices. Also, another thing is that the results might not come up straight away so give it some time. So from the two IP addresses that I have found I run a scan on one of them and this is the result:

```
Discovered open port 135/tcp
Discovered open port 25/tcp
Discovered open port 143/tcp
```

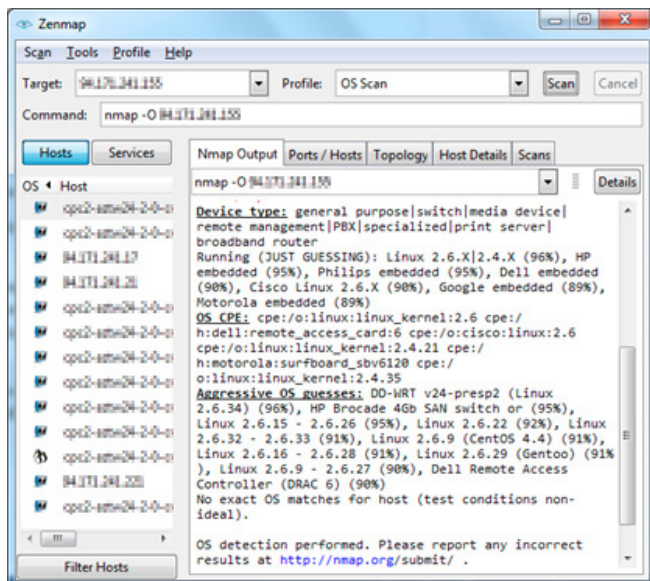


Figure 8. NMAP OS Scan Result

```
Discovered open port 5900/tcp
Discovered open port 3389/tcp
Discovered open port 53/tcp
Discovered open port 23/tcp
Discovered open port 21/tcp
Discovered open port 110/tcp
Discovered open port 1723/tcp
```

From the results produced, my first thought would not be – that is a mobile phone, but rather, some kind of access point that my phone is allowed to access in order to get the internet and get mail. We can see SMTP, IMAP, VNC, Microsoft Terminal Server and many other ports registered on this host. I am sure that no iPhone on the planet is running Microsoft Terminal Server for obvious reasons! Although to my surprise, the NMAP `-O` IP command brought rather satisfying results. Perhaps the iPhone was jailbroken.

```
Device type: media device|phone|general purpose|firewall
Running (JUST GUESSING): Apple iOS 5.X|4.X (98%),
FreeBSD 6.X (89%), Netasq embedded (89%)
```

During this scan I discovered another useful option which would be to skip PING in the first part of the scan as some devices won't accept it due to the configuration. If it isn't pingable it does not mean that the device is down. NMAP also has another trick up its sleeve which is an option called `-Pn` which will carry on scanning even if the device does not respond to PING commands.

The conclusion here is that no matter what network situation you put NMAP through it seems to cope with it rather well and it always will turn out some results. Please do not turn off your moral compass before or after your scans and do not take advantage of other users running poorly configured networks.

JAKE WYLEZEK



Jake Wylezek is a final year student of Computer Security at University of West of England (Bristol, United Kingdom), founder of Obyte Hosting Services and works for Hewlett-Packard. When he isn't glued to the keyboard, he spends time in the garage working on his beloved cars. He recently won the i43 Multi-play PC Speed Building Challenge. He can be contacted at jakub@wylezek.com.

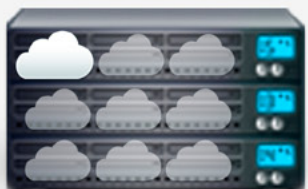
He recently won the i43 Multi-play PC Speed Building Challenge. He can be contacted at jakub@wylezek.com.



is a hosting service company. With products varying from managed hosted services and applications through to the ability to co-locate with us the customer will have the right options available to them. With a kind customer facing side and a multi-skilled team we offer everything from advise to fully virtual solutions.

Virtual Private Servers

Virtual Dedicated Servers



Server Colocation

Web Hosting



Services starting at **£2.69** per month

Promo code **5% off: VPSPROMO**
Subject to availability.

Contact us:
info@0byte.co.uk
www.0byte.co.uk



Follow us:
[@0byteServices](https://twitter.com/0byteServices)

Nmap – The Swiss Army Knife of Network Discovery

Nmap is a popular free and open source port scanner if you have not heard of it. It is mentioned frequently in Hakin9 and other online articles, and also featured as the hacker's choice of tool in several movies [1]. You can use Nmap to scan entire networks with a simple line of command or just an individual host. To the casual observer, Nmap is just a network port scanner. However, it is a powerful toolkit comprised of many useful utilities (commands and GUI).

In this short article, I will introduce to you the “*nmap*” command and the scripting engine that is really useful to know in order to do ‘almost anything’ with Nmap. The flow of this article is to start from the very basic and then incrementally introduce capabilities of Nmap with real life demonstrations that are safe to execute. And sprinkled around in this article footnotes are links back to the Nmap site so that you can further your research from there.

At the end of this simple how to, I hope you will like Nmap and use it in many ways possible for your mission.

In short, what you will learn from this article:

- Installing Nmap.
- Command line and GUI usage.
- Nmap Scripting Engine (NSE) and web application testing usage (discovery).

Requirements

To follow along, you should be comfortable with using the command line on a UNIX environment but don't worry, you won't be doing anything more than a single line of command each time and that is minimal. Nmap comes with a very nice GUI and you can do pretty much everything with it. The GUI comes with context sensitive help for every option on Linux, Windows and Mac OS X. I will come to

that later with examples and screen captures to let you preview the simplicity of the GUI.

To increase the value of using Nmap, the LUA programming language [2] is used to write NSE scripts. There are more than 400+ scripts included in the Nmap installation, and that's a lot of references and knowledge to leverage upon for learning and customizing your own script. You can al-

```
Nmap scan report for 172.16.117.128
Host is up (0.0032s latency).
Not shown: 65529 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
5800/tcp  open  vnc-http
| my-http-strangeport:
|   Server: TigerVNC/4.0
|   Date: Thu, 02 May 2013 10:29:45 GMT
|   Last-Modified: Thu, 02 May 2013 10:29:45 GMT
|   Connection: close
|   Content-Type: text/html
|
| (Webservice spotted - 172.16.117.128:5800)
5900/tcp  open  vnc
7711/tcp  open  unknown
| my-http-strangeport:
|   Date: Thu, 02 May 2013 10:29:45 GMT
|   Server: Apache
|   Last-Modified: Fri, 15 Mar 2013 11:27:00 GMT
|   ETag: "32f-4d7f4eae9c2be"
|   Accept-Ranges: bytes
|   Content-Length: 815
|   Vary: Accept-Encoding,User-Agent
|   Connection: close
|   Content-Type: text/html
|
| (Webservice spotted - 172.16.117.128:7711)
Nmap done: 1 IP address (1 host up) scanned in 38.68 seconds
```

Figure 1. Hunting for hidden web services

so use shell scripting in the command line to pipe or chain nmap commands with other tools too, to achieve desired outcomes.

And if you just want to try Nmap without messing up your current machine, you can grab a copy of Backtrack live CD. It contains many tools, including Nmap preinstalled, and you can run it either as a virtualized guest OS on top of your OS or boot it live from your machine.

Because port scanning is all about the network, having some understanding how networking works is going to help you figure out how to use Nmap and read its results. If not, hopefully this article provides you the basics to further your own reading.

Installation

Visit Nmap's download page [3] and get the latest version (e.g. 6.25). Binaries are available so you don't have to compile from source. Download and follow the installation steps provided by the installers. Take note where nmap is installed to.

It is also very simple to compile from source if there is not a suitable binary for your OS or the nmap version that comes with the OS is not the latest version. Simply download the source bzip/gzip, extract it and then `cd <extracted directory>; ./configure ; make` and the binary is compiled and ready to use. You can either use Nmap straight inside the directory e.g. `<path to extracted nmap/`

```
nmap version 6.2009W | http://nmap.org |
Platform: i686 pc linux gnu
Compiled with: nmap-liblua-5.2.1 openssl-1.0.1e libpcap-0.30 libcap-1.3.0 nmap-libnet-1.12 ipv6
Compiled without:
Hostable ports engines: snmp snmp-select
```

Figure 2. nmap version testing

```
Nmap 6.20SVN | http://nmap.org |
Usage: nmap [Scan Type(s)] [Options] (target specification)
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilenames>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
```

Figure 3. nmap options (sample)

```
NMAP(1)                                Nmap Reference Guide                                NMAP(1)
NAME
  nmap - Network exploration tool and security / port scanner
SYNOPSIS
  nmap [Scan Type...] [Options] (target specification)
DESCRIPTION
  Nmap ("Network Mapper") is an open source tool for network exploration
  and security auditing. It was designed to rapidly scan large networks,
  although it works fine against single hosts. Nmap uses raw IP packets
  in novel ways to determine what hosts are available on the network,
  what services (application name and version) those hosts are offering,
  what operating systems (and OS versions) they are running, what type of
  packet filters/firewalls are in use, and dozens of other
  characteristics. While Nmap is commonly used for security audits, many
  systems and network administrators find it useful for routine tasks
  such as network inventory, managing service upgrade schedules, and
  monitoring hosts up/down status.
  A typical Nmap scan is shown in Example 1. The only Nmap arguments used
  in this example are -A, to enable OS and version detection, script
  scanning, and traceroute; -T4 for faster execution; and then the two
  target hostnames.
  Example 1. A representative Nmap scan
  # nmap -A -T4 scanme.nmap.org
  Nmap scan report for scanme.nmap.org (74.207.244.221)
```

Figure 4. nmap man documentation (sample)

nmap> or install Nmap to your OS by additional `su root; make install` so that it can be used simply by calling `nmap` anywhere in the OS. Test if nmap is installed fine by `nmap -v` and if there's any error indicating otherwise, before continuing with the How To.

Note: if you are already running your Linux/Mac OS X shell session as `root` user, you can omit "sudo" in all the commands since you are already "root" with all the privileges.

Port Scanning 101 (Command Line)

Running "nmap" without any arguments (Figure 3) shows all the options you can use with some descriptions and "man nmap" (Figure 4) on Linux or Mac OS the more descriptive off-line documentation. Of course everything else you want to find out about Nmap is on the `nmap.org` website.

Running nmap is very straightforward. Most people would just run `nmap <target hostname/ip` without any option and it will still return results (limited) and can call it a day. Nmap is not like the `ping google.com` to find out if the Internet's down or not when you can't surf the web. It is much more. For example, behind each open port is a potential backdoor to somewhere else.

If you are observant enough of the options and man page, there's an interesting hostname mentioned in Figure 3 and 4. It's not a random hostname. Open url `http://scanme.nmap.org` using a web browser and understand [4] that you are authorized to perform scans against this host without getting into any trouble. Port scanning might not be legal in certain places but scanme.nmap.org gave the permission so I shall use it as the target for the examples, and you can do that too.

```
Nmap scan report for scanme.nmap.org (74.207.244.221)
Host is up (0.28s latency).
PORT      STATE SERVICE
80/tcp    open  http
Nmap done: 1 IP address (1 host up) scanned in 5.91 seconds
```

Figure 5. Test scanme.nmap.org port 80

```
Nmap scan report for scanme.nmap.org (74.207.244.221)
Host is up (0.29s latency).
Not shown: 987 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    filtered smtp
80/tcp    open  http
135/tcp   filtered msrpc
139/tcp   filtered netbios-ssn
445/tcp   filtered microsoft-ds
593/tcp   filtered http-rpc-epmap
1026/tcp  filtered LSA-or-nterm
1027/tcp  filtered IIS
4444/tcp  filtered krb524
6129/tcp  filtered unknown
6667/tcp  filtered irc
9929/tcp  open  nping-echo
Nmap done: 1 IP address (1 host up) scanned in 9.34 seconds
```

Figure 6. Test scanme.nmap.org without port restriction

Since this hostname is serving off a web page, TCP port 80 (HTTP) should be opened. So let's execute `nmap -p 80 scanme.nmap.org` to start the first simple port scan. This command tells nmap to scan only port 80 [5] and target scanme.nmap.org. The result (Figure 5) showed that the TCP port 80 is indeed open and recognized as the HTTP service.

If there is no restriction of the port to scan, i.e. you wished to do a discovery scan of what ports are opened or closed on the target, omit the `-p 80` argument and the results (Figure 6) will output more or less information depending on the situation (e.g. are there any firewalls protecting the target etc.).

The speed of scanning as you can see is pretty fast. Just a couple of seconds and results are back. The default Nmap scan does 1000 ports for each protocol (TCP in this case). Without 'root' privileges, Nmap defaults to the CONNECT scan technique (sT). If you change the test to use SYN scan (sS), the results will be much faster but access to admin rights is necessary, thus the `sudo` prefix to the commands.

```
Stats: 0:00:01 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 0.04% done
Stats: 0:00:02 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 0.07% done
Stats: 0:00:05 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 1.11% done; ETC: 14:35 (0:07:24 remaining)
Stats: 0:00:10 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 1.40% done; ETC: 14:39 (0:11:15 remaining)
```

Figure 7. Pressing RETURN key when a scan is in progress on a CONNECT scan technique

```
Initiating Ping Scan at 14:55
Scanning scanme.nmap.org [74.207.244.221] [4 ports]
Completed Ping Scan at 14:55, 0.29s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 14:55
Completed Parallel DNS resolution of 1 host. at 14:55. 0.02s elapsed
Initiating SYN Stealth Scan at 14:55
Scanning scanme.nmap.org [74.207.244.221] [65535 ports]
Discovered open port 22/tcp on 74.207.244.221
Discovered open port 80/tcp on 74.207.244.221
SYN Stealth Scan Timing: About 3.24% done; ETC: 15:11 (0:15:25 remaining)
SYN Stealth Scan Timing: About 5.5% done; ETC: 15:14 (0:17:22 remaining)
SYN Stealth Scan Timing: About 8.51% done; ETC: 15:13 (0:16:19 remaining)
SYN Stealth Scan Timing: About 10.04% done; ETC: 15:15 (0:18:04 remaining)
```

Figure 8. Verbose scan on a SYN scan technique

```
SYN Stealth Scan Timing: About 94.60% done; ETC: 15:20 (0:01:19 remaining)
Completed SYN Stealth Scan at 15:20, 1498.78s elapsed (65535 total ports)
Nmap scan report for scanme.nmap.org [74.207.244.221]
Host is up (0.29s latency).
Not shown: 65213 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    filtered smtp
80/tcp    open  http
135/tcp    filtered esrpc
137/tcp    filtered netbios-ns
139/tcp    filtered netbios-dgm
139/tcp    filtered netbios-ssn
445/tcp    filtered microsoft-ds
593/tcp    filtered http-rpc-epmap
1026/tcp   filtered LSA or nterp
1027/tcp   filtered IIS
2745/tcp   filtered unknown
3127/tcp   filtered unknown
4444/tcp   filtered krb524
5553/tcp   filtered cgm-eventannd
5554/tcp   filtered sql-esphttp
6129/tcp   filtered unknown
6667/tcp   filtered irc
6899/tcp   filtered unknown
8929/tcp   open  nping-echo
17300/tcp  filtered kuang2
47165/tcp  filtered unknown

Read data files from: /usr/local/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 1499.21 seconds
Raw packets sent: 67648 (2.976MB) | Rcvd: 66607 (2.668MB)
```

Figure 9. Scanning all ports ("`-p-`")

Earlier scans showed very clearly whether a port was open or closed. This new scan showed "filtered" ports and there are quite a number of them e.g. port 25 (SMTP). If you were to execute `telnet scanme.nmap.org 25`, the connection does not seem to be able to be established, versus if you were to `telnet scanme.nmap.org 80` or `telnet scanme.nmap.org 22` (exit by `CTRL+J -> quit -> RETURN`) a connection can be established. Why? From the Nmap Reference Guide [6], "filtered" means the port cannot be determined in "open" state.

Let's assume there might be a bunch of other ports on the host and do scan on all of them by using `nmap -p- scanme.nmap.org` (ports 1 through 65535, will take a very long time). Wait for a couple of seconds, without observing network packets using e.g. ethereal/Wireshark [7], it is not easy to know whether the scan is still in progress or not. Try to press the RETURN key at any point and you can see the ongoing scan progress with estimated time to complete (Figure 7). This is a helpful (progress) indicator albeit manual.

However, pressing RETURN will return 'empty' with `sudo nmap -sS -p- scanme.nmap.org`. To add more information to the scan progress, use the `-v` (verbosity) argument instead e.g. `sudo nmap -sS -p- -v scanme.nmap.org` (Figure 8). Now there will be updates as to how the scan is progressing and when it will complete.

Using sniffers like Wireshark to monitor the interface is valuable, to know what is going on during a network task e.g. port scanning. There are Nmap features to "decoy" self using 'zombies'

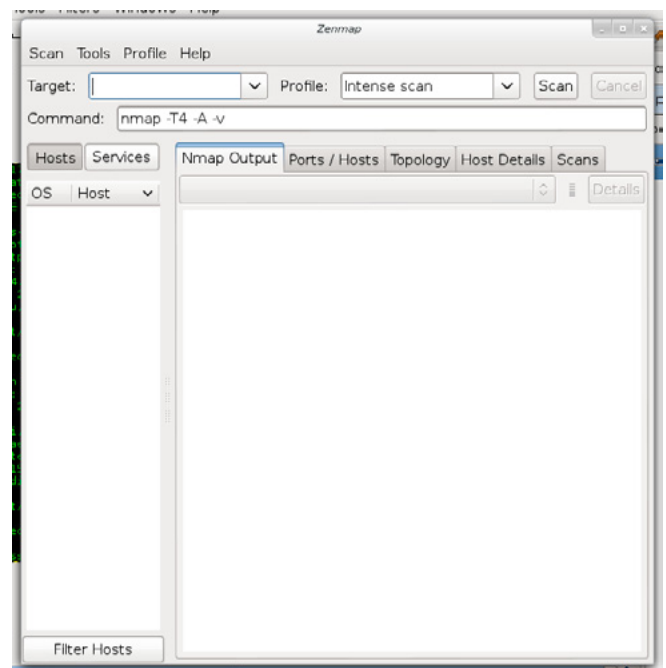


Figure 10. zenmap launched


```
james@lenny:~$ nmap | grep "\-A"
-A: Enable OS detection, version detection, script scanning, and traceroute
```

Figure 11. “-A” argument

(Idle [8] scanning) and also methods to use a proxy such as Tor so to not reveal your identity to the targets during scanning. You should use Wireshark and analyze the traffic during testing when possible to confirm there are no leakages before scanning on actual targets.

Figure 9 shows the end result of the `-p-` (all ports) scan after a couple of minutes against `scanme.nmap.org` from my location.

Zenmap (GUI)

You can launch Zenmap from the application launcher from the OS GUI or execute “`sudo zenmap`” from the command line (Figure 10).

It offers the “intense scan” profile with the arguments `-T4 -A -v`. Besides `-v` which is already demonstrated, `-T` means timing (not the actual measure of time) on the scale 0-5 (“higher is faster”). `-T3` is the default if not specified as mentioned in the Timing and Performance [9]. `-A` on the other hand is a combination of “Enable OS detection, version detection, script scanning, and traceroute” (Figure 11). Script scanning is an interesting feature, will get to it later.

Let’s run the scan with the target `scanme.nmap.org`. Figure 12 to 15 show the different views of the same scan. “Nmap Output” (Figure 12) is the same

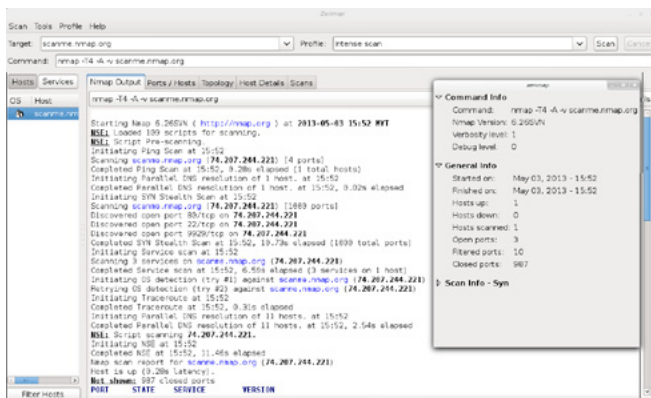


Figure 12. Zenmap: nmap output

Port	Protocol	State	Service	Version
22	tcp	open	ssh	OpenSSH 5.3p1 Debian 3ubuntu7 (Ubuntu Linux; protocol 2.0)
25	tcp	filtered	smtp	
80	tcp	open	http	Apache httpd 2.2.14 ((Ubuntu))
135	tcp	filtered	msrpc	
139	tcp	filtered	netbios-ssn	
445	tcp	filtered	microsoft-ds	
593	tcp	filtered	http-rpc-epmap	
1026	tcp	filtered	LSA-on-nterm	
1027	tcp	filtered	iis	
4444	tcp	filtered	krb524	
6129	tcp	filtered	unknown	
6667	tcp	filtered	irc	
9929	tcp	open	nping-echo	Nping echo

Figure 13. Zenmap: posts/hosts

as you would observe if you ran the same command in the command line instead.

Zenmap has a few ready-made profiles that you can click and run immediately. For the sake of understanding what each profile does, it’s better to click “*Profile > Edit Selected Profile*” or `CTRL+E` and then read the “Help” section (Figure 16) for each argument (option) interested in. You can also add new profiles by “*Profile > New Profile or Command*” or `CTRL+P`.

All profiles are saved somewhere [10] depending on your OS and you can refer to them even from the command line for quick reference e.g. `~/zenmap/scan_profile.usp` (Figure 17).

Nmap Scripting Engine (NSE)

Nmap has a very cool feature, in case you haven’t noticed, and that is the NSE. There are today more than 400 scripts, contributed by various authors, ready to use and they are all free.

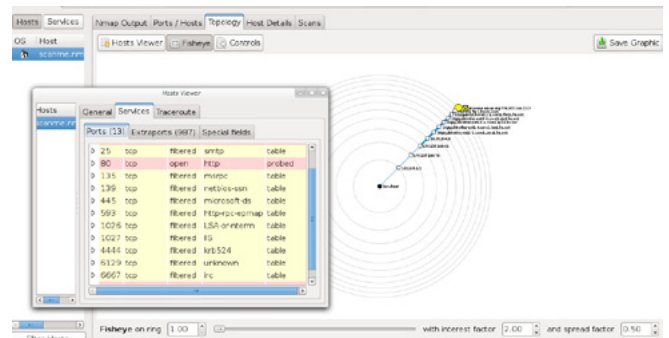


Figure 14. Zenmap: topology

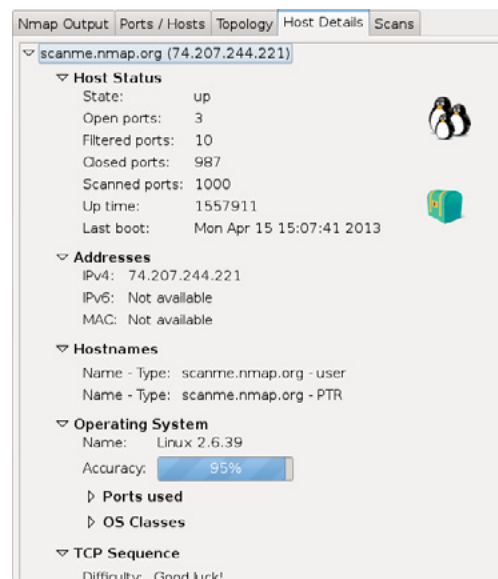


Figure 15. Zenmap: host details

Running your first NSE scan is as simple as `nmap -v -sC -sS -T4 -sC scanme.nmap.org`. What's new here is the `-sC` argument that means run the "default" set of scripts and depends on your Nmap version, run a bundle of scripts. Mine showed "NSE: Loaded 95 scripts for scanning", which is a lot of actions going on besides the usual port scanning and it took a total of 11 seconds which is not too slow at all. What is some of the additional information in the result? I'm not going to post the results here but rather prefer you go and try it yourself and find out what `-sC` output is like. If you are wondering what were all those 95 scripts ran, they are listed, categorized (e.g. "default") and explained in the NSEDOC portal [11] along with the other 400 plus scripts.

Using `-sC` is not the only way to call the NSE scripts [12]. You can call up scripts using `--script <filename>|<category>| ...` and as you had already read, `-sC` is the same as `--script default`. You will see later that it is possible to pass a NSE script to Nmap from outside the `/scripts` directory, and it is very flexible.

One problem I foresee is that if you need to use your custom script from Zenmap, the file has to be added first into the scripts directory (softlink is fine) and then `sudo nmap --script-updatedb` ran to update the scripts database. After that you can make use of the GUI Scripting feature to fill in the script argu-

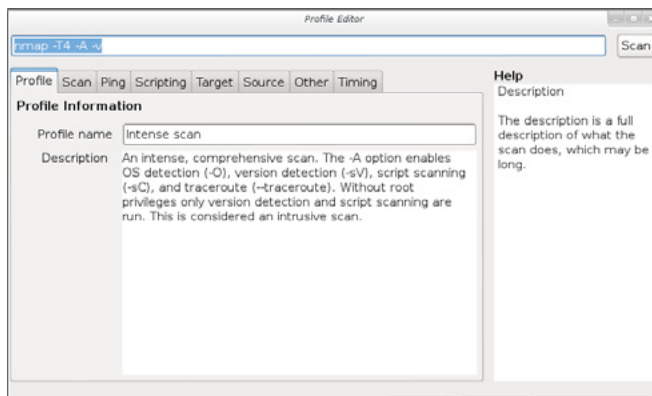


Figure 16. Zenmap: intense profile

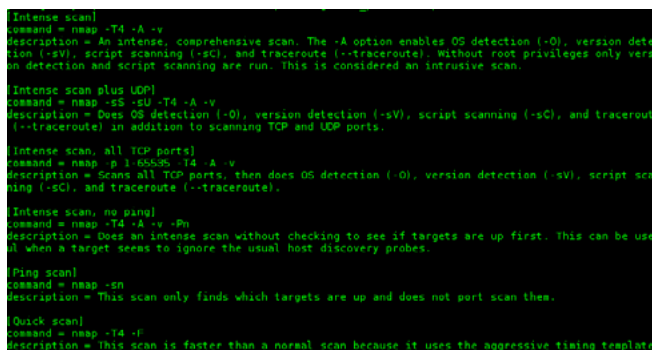


Figure 17. scan_profile.usp

ments like the other built-in scripts. Otherwise, if you can still type in those from an the command, and it will still run fine from external location without the `--script-updatedb`.

Let's take a look at one of the NSE scripts, "http-enum" [13] and demonstrate how to use it in the context of web application scanning. The documentation states that the usage is `nmap -sV --script=http-enum <target>`. So on my machine against `scanme.nmap.org` it would be `sudo nmap -sV -p 80 -v --script=http-enum scanme.nmap.org` (addition `-p 80` and `-v`. You know the purpose of these, it means to test port 80 only and be verbose in the output). And what is the result? `Scanme.nmap.org` has some interesting folders for you to find as part of experiencing Nmap for yourself.

To provide an example of doing your own custom NSE script, here is a (Figure 19) simple NSE script I had written to test if the open TCP port is also harboring a web service. The website owner perhaps wanted to cloak it by changing it to a non standard port 80/8080/443/8443 etc thinking that by doing so will make the web service more secure (but it doesn't, as I will demonstrate. The script reveals the port's true purpose in very little time in Figure 18).

What this script does is to react to ports that are both TCP and open state, then perform a HTTP GET request on the port at the root "/" path e.g.

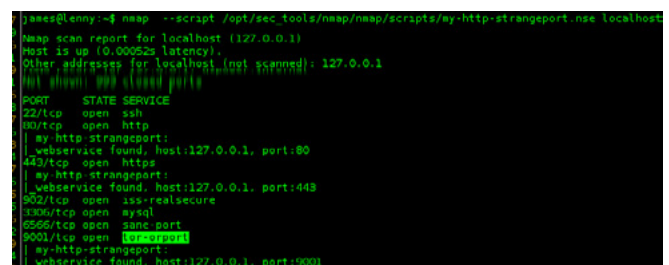


Figure 18. Web service hunting NSE script

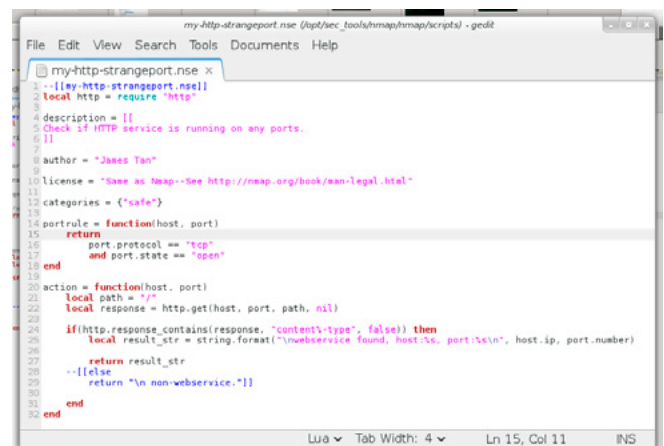


Figure 19. Writing NSE script in gedit with syntax highlighting

Practical Nmap Scanning



Network Mapper (Nmap) allows for the discovery of live computers/hosts on a network as well as detects running services and supported communication protocols. It's one of the most essential tools for any systems/network administrator, IT security professional and/or hacker. This instructional will guide you through using Nmap to effectively scan a subnet for live hosts, determine the status of firewall ports, iterate through running services and identify vulnerabilities.

The examples shown below are real results from a virtual network I've set up just for these tests. Outlined below is the layout of that network (Figure 1).

Basically, both Web01 and Sec01 have to travel through the gateway firewall/router Untangle appliance. They're all connected via a VMware vSwitch and all have full visibility to each other. As you can see, I've laid out what to expect to find within the hosts. Web01 (our victim server) is running Ubuntu 12.10 Server Edition utilizing one of the newer versions of the Linux Kernel 3.x. The only

ports exposed on our victim are 22 (OpenSSH), 80 (Apache HTTP) and 443 (Apache HTTPS).

Installation

So let's get started by actually installing Nmap from the Ubuntu software repository. If you're on another platform, consult nmap.org on how to download the source and compile it. As with most software in Ubuntu, we'll use the Aptitude package manager by running the command `$ sudo apt-get install nmap` (Figure 2). There may be a few dependencies to satisfy which will need to be accepted (by entering 'Y' at

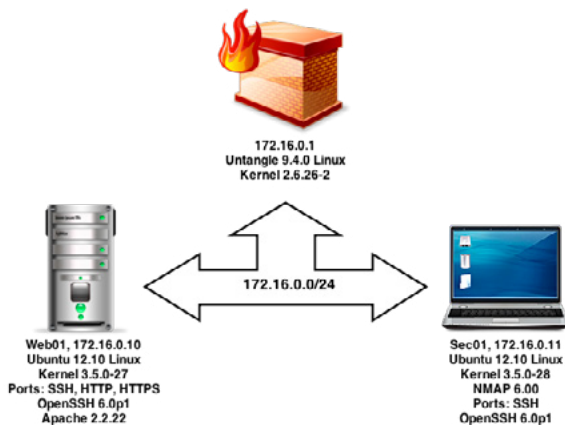


Figure 1. Sandbox network for Nmap testing. Typical SMB LAN with a Linux-based router (Untangle), Ubuntu server running a Web Server and SSH (Web01) and a "client" PC running Ubuntu (Sec01)

```
joornutt@sec01:~$ sudo apt-get install nmap
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libblas3 liblinear-tools liblinear1 liblua5.1-0
Suggested packages:
  libsvm-tools liblinear-dev
The following NEW packages will be installed:
  libblas3 liblinear-tools liblinear1 liblua5.1-0 nmap
0 upgraded, 5 newly installed, 0 to remove and 0 not upgraded.
Need to get 4,856 KB of archives.
After this operation, 16.6 MB of additional disk space will be used.
Do you want to continue [Y/n]?
```

Figure 2. Installation of Nmap using Aptitude in Ubuntu Server. If you compile Nmap from source, make sure to include OpenSSL support at compile time!

```
joornutt@sec01:~$ nmap -V
Nmap version 6.00 ( http://nmap.org )
Platform: x86_64-unknown-linux-gnu
Compiled with: liblua-5.1.5 openssl-1.0.1 libpcap-0.90 libccap-1.3.0 nmap-libdnet-1.12 ipw6
Compiled without:
```

Figure 3. Displays the Nmap version and compile-time options such as OpenSSL support

the prompt) before the Nmap installation will continue. The default Nmap package when run in Ubuntu 12.10 Server is version 6.00. If you've started this article using the latest Backtrack (version 5 R3) then your version will be at Nmap 6.10 (Figure 3).

Subnet Inventory

So how do you get started with inventorying a network and finding a victim host? Well, Nmap provides several scanning techniques to identify live hosts within a subnet (Figure 4).

-PR	ARP Ping Sends an ARP Request out on the network and waits for a reply. If it receives a reply then the host is up and Nmap will then know the IP address of the system anyways (which it also could have found using other slower IP-based scans).
-n	No DNS Resolution Tells Nmap not to attempt DNS resolution on any discovered hosts. This increases the speed of the scan. We will attempt reconnaissance on a particular host after we figure out what targets even exist.
-sn	No Port Scan This option mainly tells Nmap to perform a "Ping scan" for inventorying a network as opposed to actually attempting service/port discovery.

Our scan reported 3 live hosts on the sandbox network. This scan tells us a lot for being an entry-level scan. For starters I now know that these are virtualized servers (VMware MAC addresses were returned, unlikely to be a vanilla client system hosted on ESX) and judging by the IP layout it's a safe bet that 172.16.0.1 is a gateway device, 172.16.0.11 is our own IP and 172.16.0.11 is an unknown VMware-hosted server. The reason we performed a bare-bones ARP sweep instead of a more comprehensive port/services scan is because of the time it takes to iterate every possible host in an entire subnet. Had we scanned ports, services, etc... on an entire subnet with even a dozen hosts, the performance hit would be noticeable and it exposes our actions to more hosts than we may want. On larger corporate or government networks it's very important to find specific targets instead of doing broad sweeps and throwing everything at a subnet because you

```

jcornutt@sec01:~$ sudo nmap -PR -n -sn 172.16.0.0/24
Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-08 21:36 EDT
Nmap scan report for 172.16.0.1
Host is up (0.00018s latency).
MAC Address: 00:0C:29:98:F3:19 (VMware)
Nmap scan report for 172.16.0.10
Host is up (0.00034s latency).
MAC Address: 00:0C:29:65:A0:1C (VMware)
Nmap scan report for 172.16.0.11
Host is up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 10.26 seconds
jcornutt@sec01:~$
    
```

Figure 4. ARP Ping scan of the sandbox network using no port scanning or DNS resolution; discovery only

never know what device(s) may be scanning or logging your actions. Keep it very simple and as normal looking as possible until a host/target is discovered.

Port Scanning

Once you've discovered a host system that you'd like to probe further, the next logical step is to see what it exposes to remote systems, after all, we have to have some sort of entry point to exploit a host. Open ports signify that there is a listener service on the host that will accept input from remote systems on the network. We need to find out what ports are open and what services are on the other side of the firewall. To do this we use some very powerful options for port analysis (Figure 5).

-sS	TCP SYN Scan This is actually the default port scanning technique. If you do not have sudo/root access to the system running Nmap then you will probably have to use <code>-sT</code> (TCP Connect Scan) instead since TCP SYN Scan relies on raw packet creation. TCP SYN is the fastest port scanning option. This ends a TCP SYN packet to the host and awaits a SYN-ACK packet indicating that port is open. It should receive a RST packet if the port is closed. By default, 1,000 of the most common ports are scanned. Use <code>-F</code> to limit this to 100.
--open	Only Show Open Ports Tells Nmap to only show us open (or possibly open) ports.
--reason	Display Port State Reason Tells Nmap to give us a reason why Nmap thinks a port is open or closed.

This shows us some very interesting ports being open. SSH, HTTP and HTTPS are shown open because of a successful TCP SYN-ACK scan. Now that we have a rough idea of what to gather intelligence on we can move forward with some more intrusive scans. Before we can really get into hardcore services detection we should find out what Nmap thinks the target Operating System (OS) is.

Operating System Detection

Finding what the target host's OS is can be critical in discovering vulnerabilities or aiding in predicting what is really happening on the local system. These options require port scanning to be enabled

```

root@sec01:/usr/share/nmap# nmap -sS --open --reason 172.16.0.10
Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-09 17:52 EDT
Nmap scan report for 172.16.0.10
Host is up, received arp-response (0.000076s latency).
Not shown: 997 closed ports
Reason: 997 resets
PORT      STATE SERVICE RFAISON
22/tcp    open  ssh     syn-ack
80/tcp    open  http    syn-ack
443/tcp   open  https   syn-ack
MAC Address: 00:0C:29:65:A0:1C (VMware)
    
```

Figure 5. TCP SYN Port Scan requesting Port State Reason

a CMS in the first place and if so, what type/version it is. To do this we need to analyze the HTTP directories returned by the Apache service. To do this we will need the aid of the Nmap Scripting Engine (NSE) which provides scriptable and modular functionality (written in the Lua programming language) that goes beyond what Nmap's core software offers. The scripting engine is one of the most powerful aspects to Nmap and you will quickly see why (Figure 8).

<code>--script= <script></code>	Specify NSE Script Use this to specify which NSE (.nse) script to run. These can be found the <Nmap root>/Scripts/ folder. Generally, in Linux, the install path for Nmap is to /usr/local/share/nmap or /usr/share/nmap.
<code>http-enum.nse</code>	[Script] Enumerate Website Directories This script uses the http-fingerprints database in Nmap to expose a list of website directories and then identify what web services are really running.
<code>http-php-version.nse</code>	[Script] Displays PHP Version This script attempts to discover the PHP version of a web host. It uses some interesting methods of discovery such as the PHP "Easter eggs" like http://<target>/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000, http://<target>/?=PHPE9568F36-D428-11d2-A769-00AA001ACF42 and HTTP header examination.

Bingo! Nmap enumerated visible directories on the web server and found that there are some very well-known directories found that link this server to the popular CMS Wordpress. Now instead of just having OpenSSH and Apache to scan for vulner-

```
root@sec01:/usr/share/nmap# nmap --script=http-enum 172.16.0.10
Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-09 17:04 EDT
Nmap scan report for 172.16.0.10
Host is up (0.000079s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
443/tcp    open  https
| http-enum:
| /wp-login.php: Possible admin folder
| /robots.txt: Robots file
| /phpmyadmin/: phpMyAdmin
| /wp-login.php: Wordpress login page.
MAC Address: 00:0C:29:65:A0:1C (VMware)
```

Figure 8. Enumerate visible web server directories and attempt to identify popular CMS's based on results

```
root@sec01:/home/jcornutt# nmap --script http-wordpress-enum 172.16.0.10
Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-09 20:33 EDT
Nmap scan report for 172.16.0.10
Host is up (0.000085s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
443/tcp    open  https
| http wordpess enum:
| Username found: sec01_wp_admin
| Username found: jtest_user
```

Figure 9. Enumerate usernames associated with a web server running Wordpress CMS

abilities we have now identified two additional potentially vulnerable services; Wordpress and PHP-MyAdmin. The PHP version is also now known to us. Older PHP versions have serious vulnerabilities that could then be exploited.

Wordpress Username Enumeration

Now that we know what we're up against we can try to actually gather some user information. In this case, we're going after usernames. These can be hidden from plain sight depending on what Wordpress theme or settings are being used so sometimes finding a username worth brute-forcing can be difficult. Luckily, Nmap has our back on this with the helpful http-wordpress-enum.nse to enumerate Wordpress usernames (Figure 9).

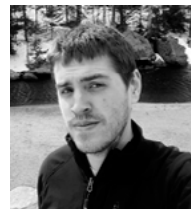
<code>--script= <script></code>	Specify NSE Script Use this to specify which NSE (.nse) script to run. These can be found the <Nmap root>/Scripts/ folder. Generally, in Linux, the install path for Nmap is to /usr/local/share/nmap or /usr/share/nmap.
<code>http-wordpress-enum.nse</code>	[Script] Enumerate Website Directories This script attempts to identify Wordpress usernames throughout a blog.

Fin.

Give yourself a big pat on the back. Using a single network scanning tool you have learned how to take a hack from identifying a target including its operating environment, enumerating open ports, detecting known services and their respective versions, using HTTP requests to find web services hosted on Apache and performed a minor exploit to identify previously unknown user accounts.

You can find more NSE scripts that are bundled with modern Nmap installs by visiting <http://nmap.org/nsedoc/>.

JOSHUA CORNUTT



Joshua Cornutt has been working in the IT industry for over 8 years. Josh started off as an entry-level computer technician fresh out of high school, ran the IT department of a local college, started, and later sold, his own IT consulting business and currently

works as a senior level systems administrator for a high-tech startup. Josh acquired skills such as C, C++ and x86 assembly programming through OS development and has worked extensively with network analysis tools such as Nmap, Cain & Abel, Nessus and Metasploit. Josh is currently the owner Joscorm.com, a technical blog that features in-depth tutorials on how to maximize security in today's world.

Nmap – The Multitool of Network Discovery

Nmap (Network Mapper) is a free-ware utility for Network scanning and security auditing. It was designed for large networks, but works on single hosts as well. It runs on all major Operating Systems and in addition to the classic command-line Nmap executable, it also includes an advanced GUI and results viewer (Zenmap). Now that you have some background information, let's jump right in!

THIS IS FOR INSTRUCTIONAL PURPOSES ONLY. THE AUTHOR IS NOT RESPONSIBLE FOR WHAT SOMEONE DOES WITH THIS INFORMATION.

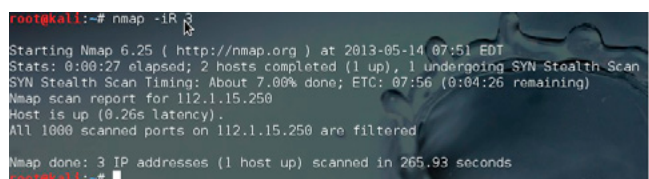
Usage: nmap [Scan Type(s)] [Options]
{target specification}

Target Specification

In this section, you give the target (or targets) that you want to scan.

Ex: google.com

- `-iL <inputfilename>`: Here you can specify a list of hosts or networks to scan.
- `-iR <num hosts>`: For Internet-wide surveys and other research, you may want to choose targets at random. The `<num hosts>` argument tells Nmap how many IPs to generate. The argument 0 can be specified for a never-ending scan (Figure 1).



```
root@kali:~# nmap -iR
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-14 07:51 EDT
Stats: 0:08:27 elapsed; 2 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 7.00% done; ETC: 07:56 (0:04:26 remaining)
Nmap scan report for 112.1.15.250
Host is up (0.26s latency).
All 1080 scanned ports on 112.1.15.250 are filtered
Nmap done: 3 IP addresses (1 host up) scanned in 265.93 seconds
root@kali:~#
```

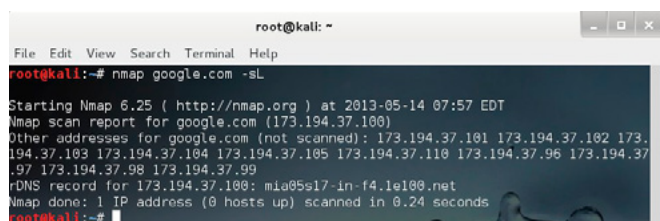
Figure 1. Target specification

- `--exclude <host1>,<host2>,[...]`: Specifies a comma-separated list of targets to be excluded from the scan even if they are part of the overall network range you specify.
- `--excludefile <exclude_file>`: This is basically the same thing as the above. Except you can exclude excluded targets are provided in a newline-, space-, or tab-delimited `<exclude_file>` rather than on the command line.

Host Discovery

In this section, you decide how you want to scan the selected host(s). You might just be looking to see if a certain host(s) are up. Or you might want to see what services are being run on a host(s). Either way, you can use Nmap to do it! There are a lot of options, so I will only go over the more important ones.

- `-sL`: This simply lists all the hosts on the network specified. Since the idea is to simply print



```
root@kali:~# nmap google.com -sL
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-14 07:57 EDT
Nmap scan report for google.com (173.194.37.100)
Other addresses for google.com (not scanned): 173.194.37.101 173.194.37.102 173.194.37.103 173.194.37.104 173.194.37.105 173.194.37.110 173.194.37.96 173.194.37.97 173.194.37.98 173.194.37.99
rDNS record for 173.194.37.100: ml-a05s17-in-f4.1e100.net
Nmap done: 1 IP address (0 hosts up) scanned in 0.24 seconds
root@kali:~#
```

Figure 2. Host discovery

a list of target hosts, options for higher level functionality such as port scanning, OS detection, or ping scanning cannot be combined with this (Figure 2).

- `-sn`: This option tells Nmap not to do a port scan after host discovery, and only print out the available hosts that responded to the scan. This is often called a ping sweep, and is more reliable than pinging the broadcast address because many hosts do not reply to broadcast queries. The default host discovery done with `-sn` consists of an ICMP echo request, TCP SYN to port 443, TCP ACK to port 80, and an ICMP timestamp request by default.

Let's break down what a TCP SYN is. TCP is a protocol used along with the Internet Protocol (IP) to send data in the form of message units between computers over the Internet. A TCP SYNchronize packet is used to initiate a TCP session. First, computer A sends a SYN to computer B. Computer B sends a SYN-ACKnowledgment packet back (to synchronize and then acknowledge the synchronization). Then Computer A sends an ACK packet to Computer B. The computers are now connected. This is called a "3-Way Handshake" and is used every time a computer connects to another computer using the TCP protocol.

- `-Pn`: This option skips the Nmap discovery stage altogether. Normally, Nmap only performs heavy probing such as version detection, or OS detection against hosts that are found to be up. Using `-Pn` causes Nmap to attempt the requested scanning functions against every target IP address specified. In this example, I used OS detection and Version detection as the requested functions (the `-A` parameter). I also used `-n` (discussed below) to make the scan faster (Figure 3).
- `-PS <port list>`: This option sends an empty TCP packet with the SYN flag set. The default destination port is 80 but alternate ports can be set with the `<port list>` parameter. Note that there can be no space between `-PS` and the port list. For example, you may use `-PS-25,80,113` but `-PS 25,80,113` would not work. The SYN flag suggests to the remote system that you are attempting to establish a connection. Normally the destination port will be closed, and a RST (reset) packet sent back.

If the port happens to be open, the target will take the second step of a TCP three-way-handshake by responding with a SYN-ACK TCP packet. The

machine running Nmap then tears down the connection by responding with a RST rather than sending an ACK packet, which would complete the three-way-handshake and establish a full connection. The RST packet is sent by the kernel of the machine running Nmap in response to the unexpected SYN/ACK, not by Nmap itself. This is used see if the target is up. A common type of firewall uses stateful rules that drop unexpected packets. This feature was initially found mostly on high-end firewalls, though it has become much more common over the years. A SYN probe is more likely to work against such a system, as unexpected ACK packets are generally recognized as bogus and dropped. A solution to this quandary is to send both SYN and ACK probes by specifying `-PS` and `-PA` (Figure 4).

```

File Edit View Search Terminal Help
root@kali:~# nmap google.com -Pn -sV -A -n
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-15 21:59 EDT
Nmap scan report for google.com (173.194.37.7)
Host is up (0.019s latency).
Other addresses for google.com (not scanned): 173.194.37.8 173.194.37.9 173.194.37.14 173.194.37.0 173.194.37.1 173.194.37.2 173.194.37.3 173.194.37.4 173.194.37.5 173.194.37.6
Not shown: 982 filtered ports
PORT      STATE SERVICE      VERSION
25/tcp    open  smtp?
|_smtp_commands: Couldn't establish connection on port 25
80/tcp    open  http?
|_http_methods: No Allow or Public header in OPTIONS response (status code 405)
81/tcp    open  hosts2-ns?
82/tcp    open  xfer?
110/tcp   open  pop3?
119/tcp   open  rntp?
143/tcp   open  imap?
|_imap_capabilities:
|_ERROR: Failed to connect to server
443/tcp   open  https?
|_http_methods: No Allow or Public header in OPTIONS response (status code 405)
|_http-robots.txt: 233 disallowed entries (15 shown)
    
```

Figure 3. Quick scanning

```

File Edit View Search Terminal Help
root@kali:~# nmap google.com -PS-80
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-14 08:15 EDT
Stats: 0:09:49 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 92.35% done; ETC: 00:26 (0:00:47 remaining)
Nmap scan report for google.com (173.194.37.14)
Host is up (0.029s latency).
Other addresses for google.com (not scanned): 173.194.37.0 173.194.37.1 173.194.37.2 173.194.37.3 173.194.37.4 173.194.37.5 173.194.37.6 173.194.37.7 173.194.37.8 173.194.37.9
DNS record for 173.194.37.14: mia05s08-in-f14.1e100.net
Not shown: 998 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https
Nmap done: 1 IP address (1 host up) scanned in 757.23 seconds
root@kali:~#
    
```

Figure 4. Probing with `-pe` and `-pa`

```

File Edit View Search Terminal Help
root@kali:~# nmap google.com --traceroute
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-14 08:29 EDT
Nmap scan report for google.com (74.125.229.206)
Host is up (0.027s latency).
Other addresses for google.com (not scanned): 74.125.229.192 74.125.229.193 74.125.229.194 74.125.229.195 74.125.229.196 74.125.229.197 74.125.229.198 74.125.229.199 74.125.229.200 74.125.229.201
DNS record for 74.125.229.206: mia04s08-in-f14.1e100.net
Not shown: 998 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https
TRACEROUTE (using port 80/tcp)
HOP RTT ADDRESS
1 0.59 ms (192.168.2.1)
2 ... 30
Nmap done: 1 IP address (1 host up) scanned in 177.40 seconds
root@kali:~#
    
```

Figure 5. Example results

- `--traceroute`: Traceroutes are performed post-scan using information from the scan results to determine the port and protocol most likely to reach the target. It works with all scan types except connect scans (`-sT`) and idle scans (`-sI`) (Figure 5).
- `-n`: Tells Nmap to never do DNS resolution of the IP addresses it finds to be up. Since DNS can be slow this makes the scan considerably faster.
- `-ss`: SYN scan is the default and most popular scan option for good reasons. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by restrictive firewalls. It is also relatively unobtrusive and stealthy since it never completes TCP connections.
- `-sF`: The FIN scan sends a TCP FIN packet. The FIN flag is used for negotiating between both systems that the communication is over and they can drop the connection. Basically it tears down the TCP virtual connection. And it appears in the last packets of the data stream.
- `-sA`: This scan is different than the others discussed so far in that it never determines open (or even open/filtered) ports. It is used to map out firewall rulesets, determining whether they are stateful or not and which ports are filtered. The scan probe only has the ACK flag set. When scanning unfiltered systems, open and closed ports will both return a RST packet. Nmap then labels them as unfiltered, meaning that they are reachable by the ACK packet, but whether they are open or closed is unknown.
- `-sW`: the window scan is the same as ACK scan except that it exploits a detail of certain systems to differentiate open ports from closed ones. It does this by examining the TCP Window field of the RST packets returned. On some systems, open ports use a positive window size (even for RST packets) while closed ones have a zero window. So instead of always listing a port as unfiltered when it receives a RST back, Window scan lists the port as open or closed if the TCP Window value in that reset is positive or zero.

The Six Port States Recognized by Nmap

OPEN

An application is accepting TCP connections, UDP datagrams or SCTP associations on this port. Security-minded people know that each open port is an avenue for attack. Attackers and pen-testers want to exploit the open ports, while administrators try to close or protect them with firewalls without thwarting legitimate users.

CLOSED

A closed port is accessible (it receives and responds to Nmap probe packets), but there is no application listening on it. They can be helpful in showing that a host is up on an IP address (host discovery, or ping scanning), and as part of OS detection.

FILTERED

Nmap cannot determine whether the port is open because of packet filtering. The filtering could be from a dedicated firewall device, router rules, or host-based firewall software. These ports frustrate attackers because they provide so little information.

UNFILTERED

The unfiltered state means that a port is accessible, but Nmap is unable to determine whether it is open or closed. Only the ACK scan, which is used to map firewall rulesets, classifies ports into this state. Scanning unfiltered ports with other scan types such as Window scan, SYN scan, or FIN scan, may help resolve whether the port is open.

OPEN/FILTERED

Nmap places ports in this state when it is unable to determine whether a port is open or filtered. This occurs for scan types in which open ports give no response. The lack of response could also mean that a packet filter dropped the probe or any response it elicited. So Nmap does not know for sure whether the port is open or being filtered.

CLOSED/FILTERED

This state is used when Nmap is unable to determine whether a port is closed or filtered.

Port Specification And Scanning Order

- `-p <port ranges>`: Only scan specified ports.
- `-F`: Specifies that you wish to scan fewer ports than the default. Normally Nmap scans the common 1,000 ports for each scanned protocol. With `-F`, this is reduced to 100.

Firewall/Ids Evasion And Spoofing

Network obstructions such as firewalls can make mapping a network exceedingly difficult. Nevertheless, Nmap offers many features to help understand these complex networks, and to verify that everything is working as intended. It even supports mechanisms for bypassing poorly implemented defenses. In addition to restricting network activity, companies are increasingly monitoring traffic with intrusion detection systems (IDS). All of the major

IDSs have rules designed to detect scans because scans are sometimes a precursor to attacks. Many of these products have recently morphed into intrusion prevention systems (IPS) that actively block traffic deemed malicious. Unfortunately for network administrators and IDS vendors, reliably detecting bad intentions by analyzing packet data is a tough problem.

As a Network Administrator, I sometimes need to scan my Network for vulnerabilities. In doing so, I sometimes need to look at it like an attacker would. So I need to scan my network like an attacker would. So I use Nmap options that are made for Firewall/IDS evasion and spoofing. I will show you how to properly evade detection. First I will show you the options.

- `-f`: The `-f` option causes the requested scan (including ping scans) to use tiny fragmented IP packets. The idea is to split up the TCP header over several packets to make it harder for packet filters, intrusion detection systems, and other annoyances to detect what you are doing. Specify this option once, and Nmap splits the packets into eight bytes or less after the IP header. Specify `-f` again to use 16 bytes

per fragment (reducing the number of fragments). I sometimes do a scan while a sniffer like Wireshark is running, to make sure that the packets are indeed fragmented.

- `-D <decoy1>[,<decoy2>[,...]]`: Causes a decoy scan to be performed, thus their IDS might report 5–10 port scans from unique IP addresses, but they won't know which IP was scanning them and which were innocent decoys. While this can be defeated through router path tracing, response-dropping, and other active mechanisms, it is generally an effective technique for hiding your IP address.
- `-S <IP_Address>`: With this you can spoof your IP address to make it look like someone else is probing the target. Very useful if you want anonymity while scanning.

BRANDEN PAUL

I have been the Network Administrator for a banking company for 7 years now. But I started working with computers long before that. I like working with computers because there is a chance to do/learn something new everyday! There is always an opportunity to better yourself in the computer field. I have bachelors degrees in Information Technology, Computer Security and Networking.

a d v e r t i s e m e n t

JOSCOR.COM

Technology Integration Research & Development

Where only your opinion matters...

Using Nmap for Outbound Traffic Analysis

You wouldn't let your kids talk with strangers in the street, right? But if you are not analyzing the servers your users are connecting to, that's exactly what you are doing.

We all have our firewalls configured to prevent pretty much all inbound traffic (with a few exceptions), and we know what outbound traffic to allow: http, https, ssh, smtp, pop, etc. And you know that when a hacker manages to land a trojan or install a backdoor in your network, the command and control outbound traffic will be via http or https most of the time. Also, if

one of your users falls for a phishing scam, his/her outbound traffic will obviously would be http.

You should be doing outbound traffic analysis, but you cannot always catch pirate signals inside a huge stream of http and https traffic. So one extra layer of defense in depth you can implement is analyzing the servers your users are connecting to.

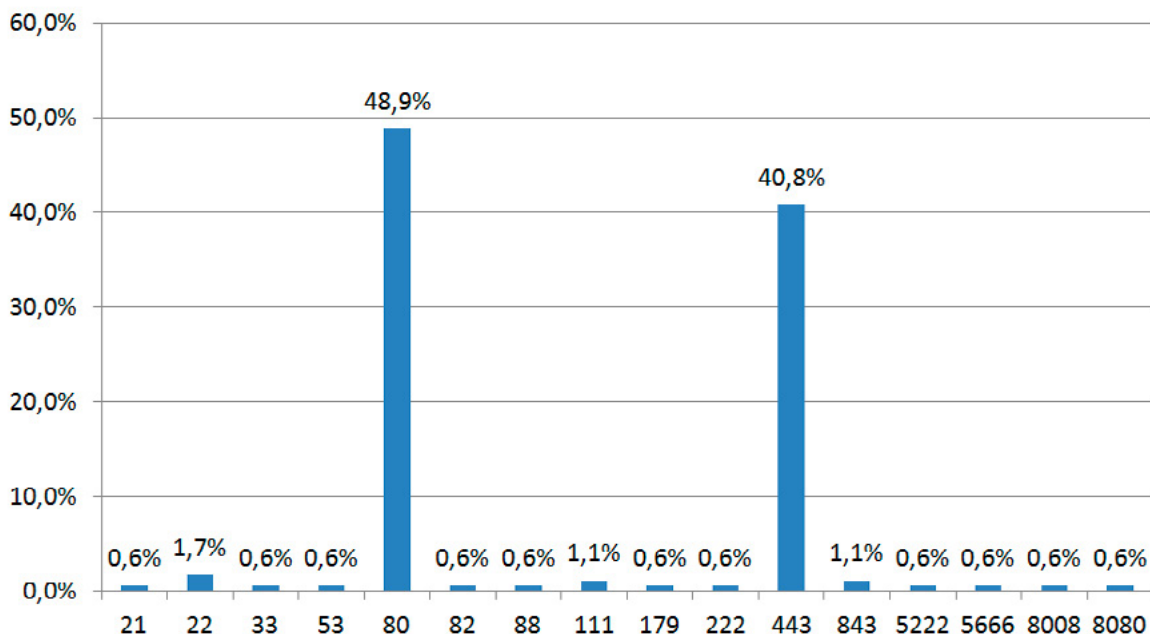


Figure 1. Port analysis Top 100 sites-SCastro



This is the hypothesis: legitimate servers are (usually) managed by smart people who have the sense to secure them. Therefore, the number of open ports will be limited, and will have primarily ports 80 and 443 open. On the other hand, derelict servers, those not being actively managed by anyone, will have all sorts of open ports, making them prime targets to be hacked and taken over, and being used as command and control hacker servers. Therefore, by doing a port scan of the servers our users are connecting to, we can deduce which connections are legitimate, and which connections may be hacker related.

Let's test the hypothesis. First, let's do an Nmap port scan of the top 100 websites. Save the list of the sites on a text file, and use the command:

```
Nmap -iL top100.txt -oX top100.xml -vv
```

With this command we are running a normal port scan, using the `-iL` command to indicate that the input list is in the file `top100.txt`, and using the `-oX` command to send the output to the new file `top100.xml`. We are saving it in xml format so we can easily open it in Excel.

Once the port scan finishes, open `top100.xml` with Excel, and do a frequency count of the ports open. You should get something like this: Figure 1.

As predicted, ports 80 and 443 are the most common ones. You can still find a few other ports open (which in strict sense should not be available on a web server), but most are well known and predictable:

- Port 21: file transfer protocol (ftp)
- Port 22: secure shell (ssh)
- Port 33: Display support protocol (dsp)
- Port 53: Domain Name System (DNS)
- Port 82: Torpark control (which probably should not be there)
- Port 88: Kerberos authentication system
- Port 111: Sun RPC
- Port 179: Border Gateway Protocol (BGP)
- Port 222: SPX Authorization Port
- Port 843: Adobe Flash
- Port 5222: Extensible Messaging and Presence Protocol (XMPP)
- Port 5666: Nagios
- Port 8008: IBM HTTP Server administration default
- Port 8080: Alternate HTTP

Now what we are going to do is scan the ports of 1000 servers known to have phishing sites, using the list available in <http://data.phishtank.com/data/online-valid.csv>.

[GEEKED AT BIRTH]



You can talk the talk.
Can you walk the walk?

[IT'S IN YOUR DNA]

- LEARN:**
- Advancing Computer Science
 - Artificial Life Programming
 - Digital Media
 - Digital Video
 - Enterprise Software Development
 - Game Art and Animation
 - Game Design
 - Game Programming
 - Human-Computer Interaction
 - Network Engineering
 - Network Security
 - Open Source Technologies
 - Robotics and Embedded Systems
 - Serious Game and Simulation
 - Strategic Technology Development
 - Technology Forensics
 - Technology Product Design
 - Technology Studies
 - Virtual Modeling and Design
 - Web and Social Media Technologies

www.uat.edu > 877.UAT.GEEK

We take that list, and clean it up in Excel to extract just the server names (remove `http://` folder paths, etc.). Then select 1,000 of those servers, save them as `.txt`, and run a port scan with

```
Nmap -iL phish.txt -oX phishresult.xml -vv
```

Needless to say, the port scan will take a while. After it's done, open the XML file with Excel, and as before, do a frequency count of the open ports. You will see that you have a lot (and I do mean a LOT) of different open ports. So let's graph the first 1,000 ports: Figure 2.

We can see spikes in predictable ports, but we can also see way too many open ports that probably should not be open:

- Port 26: Home of the Netsky worm, and used also for simple file transfer protocol, something phishers would use.
- Port 81: Topark (or xB browser), used to navigate the Thor anonymity network.
- Port 110: Post Office Protocol (POP3)
- Port 111: Sun RPC, a vulnerable protocol which should never be open to the Internet.
- Port 143: Internet Message Access Protocol (IMAP).
- Port 465: Simple Mail Transfer Protocol (SMTP).
- Port 554: Real Time Streaming Protocol (RTSP).
- Port 555: Many trojans use this port.
- Port 587: Mail submission.
- Port 873: rsync file synchronization protocol.

- Port 993: Internet Message Access Protocol over SSL (IMAPS)
- Port 995: Post Office Protocol 3 over TLS/SSL (POP3S)

Now, there's nothing wrong with most of these legitimate ports; the problem is when a web server (ports 80 and 443) also have several of these ports open. Having so many ports open increase the risk of getting hacked.

These two scans show strong evidence that there is a correlation between port hygiene and security: if a server has open ports that should not be open, it is much more likely to be an insecure server, and you should check it out in detail.

So, I suggest you run a weekly scan (or daily if you can) on the servers your network is connecting to. Anything that does not look kosher, take a closer look.

You can use Wireshark to register the endpoints your network is talking to, and then run an Nmap scan.

You can also automate this. You can write a script that picks up your firewall logs, tells Nmap to run a port scan on the servers, and send you an alert if unusual ports are open. Then you can proceed to blacklist them on your firewall.

And there you go, one more layer for your defense in depth.

SERGIO CASTRO

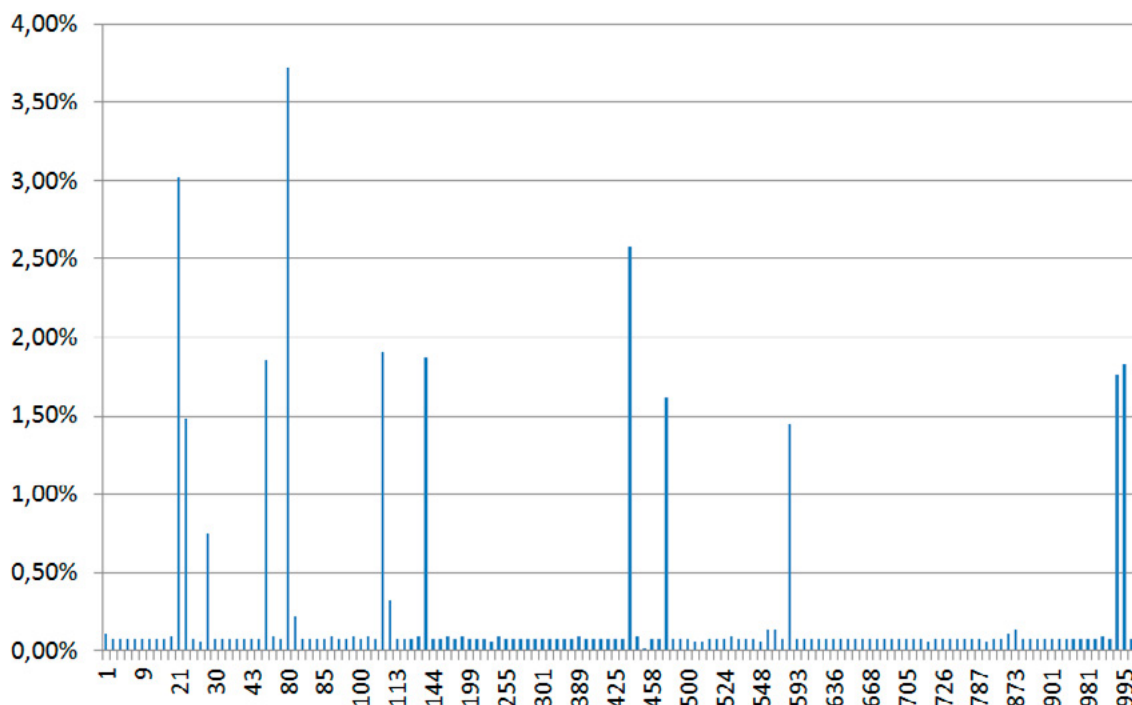


Figure 2. Sergio Castro Port Analysis



ANRC



**A Cyber criminal can target and breach
your organization's perimeter in less than
a second from **anywhere** in the world ...**

Are You Prepared?

ANRC delivers advanced cyber security training, consulting, and development services that provide our customers with peace of mind in an often confusing cyber security environment. ANRC's advanced security training program utilizes an intensive hands-on laboratory method of training taught by subject matter experts to provide Information Security professionals with the knowledge and skills necessary to defend against today's cyber-attacks and tomorrow's emerging threats.

ANRC's consulting and development services leverage team member knowledge and experience gained in the trenches while securing critical networks in the U.S. Department of Defense and large U.S. corporations. ANRC tailors these services to deliver computer security solutions specific to the needs of the customer's operational environment. Our approach emphasizes a close relationship with our clients as an integral part of our service. We believe we're all in the security battle together, and we view our customers as key members of our team in the fight.

TRAINING :: CONSULTING :: SOLUTIONS www.anrc-services.com

Refining Your Nmap Scan Strategy

Disclaimers: This is not a troll. ☺ Scans were run as root (or sudo) using Nmap version 6.25.

Few would dispute that the de facto standard scanning tool is Gordon Lyon's (aka Fyodor) Nmap. However, if we asked you to thoroughly and efficiently scan a remote class C (192.168.1.1/24) network, which of the following would you run?

- a) `nmap 192.168.1.0/24`
- b) `nmap -sS 192.168.1.0/24`
- c) `nmap -sS 192.168.1.0-255`
- d) `nmap -sS -Pn -p 0-65535 192.168.1.0/24`

The answer we hear most often is option a. While this may work for small networks, it does not scale for larger networks or more thorough assessments. The astute reader will notice that options a, b, and c, operate identically. Option a provides the network range in CIDR notation and since `-sS` is the default scan type when no options are supplied--option b is identical to option a. Examining option c, reveals that it is the same as options a and b, except that the target is supplied using a network range instead of CIDR notation.

The problem with options a, b, and c is that they will not thoroughly scan the remote class c network as they will only scan the top 1000 TCP ports. Option d is close to what we are looking for since it scans all of the TCP ports; however, it lacks ef-

ficiency since we will be scanning all ports on all hosts, including dead IP space.

The most important thing to keep in mind is that in this simple example, we are only asking to scan a class C network. However as security practitioners, we are often faced with multiple class C's, class B's or even in some cases, class A networks.

We will take the rest of this article to outline a scan strategy for networks of all sizes and some tips and tricks for carving the output.

Outline

- Host Discovery
 - `-sn` "Ping Scan"
 - Throttling Pro-tips
 - Generate Live Hosts List
- Port Discovery
 - Most Common Ports
 - Full Port Scans
- Putting it all Together
- Service and Operating System
- Nmap Scripting Engine (NSE)
- Manipulating Output
 - Carving Standard Output
 - Convert to HTML
 - Processing Script
- Summary

Host Discovery

A methodical approach to scanning usually involves performing host discovery first, which eliminates the problem we saw with the above option `d--scanning` copious amounts of dead IP space. Fortunately, Nmap is quite intelligent and flexible for performing host discovery scans. The common host discovery scan options are listed below:

```
-sn: Ping Scan - disable port scan
-PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or
                        SCTP discovery to given ports
-PE/PP/PM: ICMP echo, timestamp, and netmask
            request discovery probes
-PO[protocol list]: IP Protocol Ping
```

The option we will focus on in this article is the most popular and the default for host discovery (`-sn` “Ping Scan”).

-sn Ping Scan

One thing to note about the `-sn` “Ping Scan” option (formerly known as `-sP`) is that it does much more than just a traditional ICMP echo request. By default against a remote network the “Ping scan” consists of the following packets (Figure 1):

- ICMP echo request
- TCP SYN to 443
- TCP ACK to 80
- ICMP timestamp request

Listing 1. Example Script

```
root@DVORAK:~/scans# nmap -sn -T4 -oA Discovery
                        192.168.1.0/24

STARTING NMAP 6.25 ( HTTP://NMAP.ORG ) AT 2013-
05-13 13:21 EDT
NMAP SCAN REPORT FOR WIRELESS _ BROADBAND _
ROUTER.HOME (192.168.1.1)
HOST IS UP (0.0023S LATENCY).
MAC ADDRESS: 00:26:62:XX:XX:XX (ACTIONTEC ELEC-
TRONICS)
NMAP SCAN REPORT FOR 192.168.1.2
HOST IS UP.
NMAP SCAN REPORT FOR ANDROID.HOME (192.168.1.3)
HOST IS UP (0.025S LATENCY).
MAC ADDRESS: CC:3A:61:XX:XX:XX (UNKNOWN)
NMAP SCAN REPORT FOR COMPUTER.HOME (192.168.1.147)
HOST IS UP (0.00035S LATENCY).
MAC ADDRESS: 24:77:03:XX:XX:XX (INTEL CORPORATE)
NMAP DONE: 256 IP ADDRESSES (4 HOSTS UP)
                        SCANNED IN 4.45 SECONDS
```

When scanning a local network, Nmap automatically switches to using ARP requests. This is desirable intelligence because fewer packets are sent and it is generally more accurate. A system that blocks all ICMP traffic and filters TCP ports 80 and 443 would be invisible to the four remote discovery packets sent above. However, hosts typically respond to ARP requests because layer 2 addresses are necessary to construct packet headers (Figure 2).

For the first packet capture, the scanning host was placed on a different subnet than that of the targets--this caused Nmap to use the 4 discovery packets shown in Figure 1. For the second packet capture, the scanning host was placed in the same broadcast range and thus ARP was utilized for discovery as shown in Figure 2. Both of the packet captures were created with the same discovery scan shown Listing 1.

Options explained:

- sn = “Ping scan”
- T4 = Throttle to aggressive
- oA <basename> = Output in all three formats (normal, XML, greppable)

Throttling Pro-tips

Aggressive (`-T4`) throttling is substantially faster than the default Normal (`-T3`) throttle. We have rarely (if ever) seen Aggressive scanning crash a host or flood a network. This should be what you start with unless you know of particular hosts that are sensitive to scanning. With that said, if hosts are known to crash on simple scans or become easily flooded, avoid scanning them with `-T4` and possibly try Polite (`-T2`) throttling. Please realize that `-T2` may be up to 10 times slower than `-T3`, so be patient and only

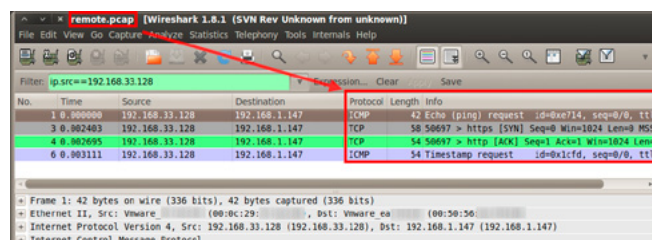


Figure 1. Packet capture confirms behavior of a remote “Ping Scan”

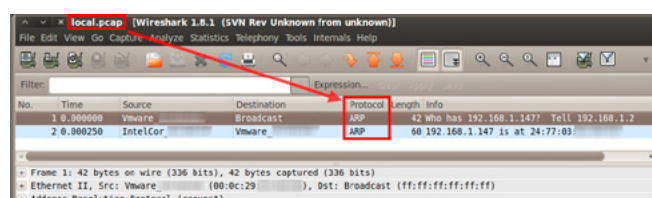


Figure 2. Packet capture confirms behavior of a local “Ping Scan”

run it on your most sensitive `hosts--not` hundreds at a time. More importantly, avoid any one-off scans such as version scanning as these are more likely to crash hosts than the speed of the scan. Some older SCADA components are known to fall over from simple port scanning, not to mention version scanning. In general, we do not recommend Insane mode (`-T5`) as this can negatively affect accuracy. Lastly, we only recommend `-T0` or `-T1` when trying to be extra stealthy (IDS evasion) and only for scanning a few ports on a few hosts because it will likely be too slow for anything else.

Generate Live Hosts List

Nmap, like other tools, accepts a file containing a list of IP addresses as input. Because the remaining scans are more intensive than the discovery scans, we will want to only feed live hosts to Nmap. We can use the output files from our discovery scan to generate this list of live hosts. Three files are generated as a result of the `-oA` option:

```
root@DVORAK:~/scans# ls
Discovery.gnmap Discovery.nmap Discovery.xml
```

Files explained

```
.nmap = Normal output (what is printed to the screen)
.gnmap = Greppable output
.xml = XML output
```

We will extract the live hosts from the `.gnmap` (grepable) file using a command such as the following:

```
root@DVORAK:~/scans# grep "Status: Up"
Discovery.gnmap | cut -f 2 -d ' ' > LiveHosts.txt
```

```
root@DVORAK:~/scans# cat LiveHosts.txt
192.168.1.1
192.168.1.2
192.168.1.3
192.168.1.147
```

Cut options explained

```
-f = field number (in this case, field 2)
-d = delimiter (in this case, a space)
```

The resulting `LiveHosts.txt` file is one IP address per line, which can be supplied as input to Nmap and other tools as well.

Port Discovery

Now that we have generated our `LiveHosts.txt` file, we can start port discovery on these hosts. If

you have many live hosts to scan and only a short amount of time, you may want to break it up into two sets of scans:

1. Most common ports
2. Full port scans

Most Common Ports

Fortunately, Fyodor performed a substantial amount of research for his Top Ports Project which he presented in 2008 at BlackHat and Decon. He performed scans on millions of Internet IPs and incorporated internal scans from large organizations to determine the most commonly open TCP and UDP ports. The key take-away from this research is that Fyodor made Nmap default port scans more efficient by targeting the top 1000 TCP and top 1000 UDP ports when no port options are provided. But just how effective is scanning only the most common ports? Here is an eye-opening table that summarizes the results of Fyodor's research:

Table 1. Summary of Fyodor's research

TCP	UDP
topports 10: 48%	topports 10: 50%
topports 50: 65%	topports 50: 86%
topports 100: 73%	-topports 100: 90%
topports 250: 83%	-topports 250: 94%
topports 500: 89%	topports 500: 97%
topports 1000: 93%	topports 1017: ~100%
topports 2000: 96%	
topports 3674: ~100%	

Source: <http://nmap.org/presentations/BHDC08/bhdc08-slides-fyodor.pdf>.

As shown in the table above, according to Fyodor's research, scanning the top 1000 TCP ports is 93% effective in identifying all open TCP ports, while scanning the top 3674 TCP ports is almost 100% effective. Additionally, scanning the top 1017 UDP ports is nearly 100% effective at identifying all open UDP ports. Since the default scan is the top 1000 TCP and UDP ports, no port options are needed below. (If scanning the top 1000 ports is not desirable, use the `--top-ports` option and specify a new number of ports to scan.)

Most common port scans

```
nmap -sS -T4 -Pn -oA TopTCP -iL LiveHosts.txt
```

```
nmap -sU -T4 -Pn -oA TopUDP -iL LiveHosts.txt
```


New Options explained

- sS = SYN Scan (half-open scanning) – Does not complete the 3-way handshake so it is faster than -sT (default scan – specifying is optional)
- sU = UDP Scan
- Pn = Do not perform discovery scan (we already know the hosts are alive)
- iL = Input from a file

Full Port Scans

Each host can have a maximum of 131,072 ports (that is $2^{16} = 65536$, then multiply by 2 for TCP and UDP). But why would anyone need to scan all of those ports? Why isn't the top 1000 good enough--especially after we highlighted how effective it was? One surprising port that is not contained in the top 1000 is TCP/2433 (the alternative default MS-SQL port, which can be a misconfigured Achilles heel for many networks). Thus, skipping the full port scan can miss critical services that may be the entry point for an attacker or a piece of malware.

Assuming that full port scans are desired, these can be executed when the most common port scans finish. Full port scans will take substantially longer to finish than the common port scans. Thus, it is ideal to have the results of the common port scans as something to review while the full port scans are running.

Note

Full port UDP scanning is VERY slow, and for extremely large networks may never finish in a reasonable amount of time.

Full port scans

```
nmap -sS -T4 -Pn -p 0-65535 -oA FullTCP -iL
    LiveHosts.txt

nmap -sU -T4 -Pn -p 0-65535 -oA FullUDP -iL
    LiveHosts.txt
```

New Options explained

- p <ports> = port list

Putting it all together

So what does the scanning process look like when combined?

Discovery

```
nmap -sn -T4 -oA Discovery 192.168.1.0/24
```

Generate Live Hosts List

```
grep "Status: Up" Discovery.gnmap | cut -f 2 -d `
    ` > LiveHosts.txt
```

Common Ports

```
nmap -sS -T4 -Pn -oA TopTCP -iL LiveHosts.txt
nmap -sU -T4 -Pn -oA TopUDP -iL LiveHosts.txt
```

Full Ports

```
nmap -sS -T4 -Pn -p 0-65535 -oA FullTCP -iL
    LiveHosts.txt
nmap -sU -T4 -Pn -p 0-65535 -oA FullUDP -iL
    LiveHosts.txt
```

Following this process should perform efficient host discovery, live host file generation, and full port scans for both TCP and UDP (along with the top 1000 ports, in case the full port scan does not finish in time).

Service and Operating System

After reviewing the common port scans (or full port scans if available), a few hosts of concern may emerge. If this is the case, a more intensive scan may be desired such as a service and/or operating system scan. These more intensive scans should not be used in discovery on a large network because it may take too long to finish. If you have many live hosts, only run these scans on select hosts where more information is desired.

Service (Version) Scan

The ports that were discovered in prior scans were quickly labeled with associated services based on the nmap-services database of about 2,200 well-known services. The example scan below shows ports 22, 23, and 80 all labeled with the expected services ssh, telnet, and http respectively.

Example SYN scan

```
nmap -sS -T4 -PN -n 192.168.1.2
PORT      STATE SERVICE
22/tcp    open  ssh
23/tcp    open  telnet
80/tcp    open  http
```

However, since applications/services can run on any arbitrary port, additional probing may be desired to ensure that the service matches the associated port. For this additional probing, version detection (-sV) is useful.

Example version scan

```
nmap -sV -T4 -PN -n 192.168.1.2
PORT STATE SERVICE VERSION
22/tcp open  ssh      OpenSSH 5.3p1 Debian 3ubuntu7
          (Ubuntu Linux; protocol 2.0)
23/tcp open  telnet   Linux telnetd
80/tcp open  ssh      OpenSSH 5.3p1 Debian 3ubuntu7
          (Ubuntu Linux; protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Notice that while the SYN scan (-ss) labeled port 80/tcp as http (per the nmap-services information), a deeper version scan (-sV) revealed that port 80 was really another instance of SSH (Figure 3).

Version scan syntax:

```
nmap -sV -T4 -Pn -oA <host>-service.txt <host>
```

New Options explained

-sV = Service Version Scan

As you can see, version scanning is quite useful, but as mentioned earlier, it comes at a cost.

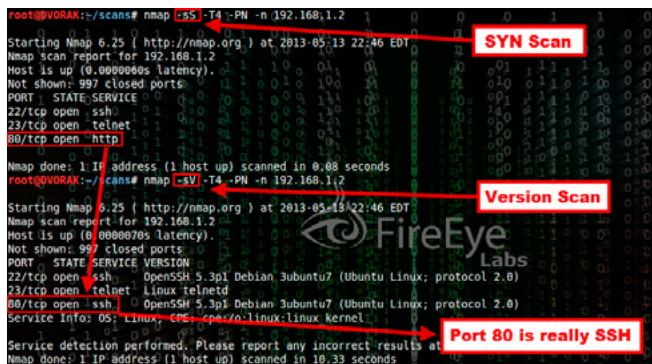


Figure 3. The screenshot above shows a SYN scan and a Version scan. The version scan revealed that port 80 is actually SSH. The version scan also correctly guessed the version of SSH

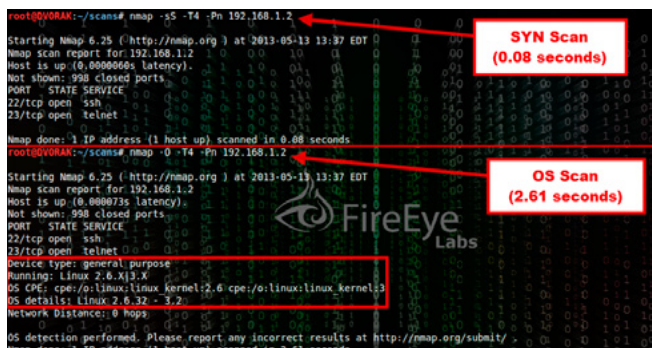


Figure 4. Comparison between SYN scan and operating system scan

It requires more packets and thus takes longer to scan. Notice in figure 3 that the SYN scan finished in 0.08 seconds, while it took the version scan 10.33 seconds. Multiply the difference by thousands of machines and it adds up.

Operating System Scan

For very large networks, the operating system for each scanned host is not always known. There are two quick checks that almost always work for high level operating system discovery (Windows vs. Linux) --especially when the two techniques are combined:

- TTL – Time To Live value (quickly obtained via ping, be sure to subtract the number of hops)
 - Windows will generally have a TTL of 128
 - Linux will generally have a TTL of 64
 - Network devices will generally have a TTL of 255
- RPC (port mapper) port
 - Windows hosts will have TCP 135 (msrpc) open
 - Linux hosts will have TCP 111 (rpcbind) open

However, if you need more detail, such as the specific version of Windows or Linux, this is where Nmap's operating system scan comes in handy. Nmap uses TCP/IP stack fingerprinting and an nmap-os-db of more than 2,600 known OS fingerprints in order to accurately identify the operating system. In most cases it is in the ballpark and if it does list a few operating systems, the correct OS is usually in the list.

```
nmap -O -T4 -Pn -oA <host>-service.txt <host>
```

New Options explained

-O = Operating System Scan

Caution: Add the necessary ports using the -p option if the ports you want to use for operating system scans are not in the top 1000 ports. Also be sure to include at least one open and one closed port to achieve a more accurate operating system guess (Figure 4).

Some may want to use both -sV and -O at the same time, which is fine. There is even a shortcut option for this discussed in the next section; however, we will provide a word of warning there as well.

NSE

Prior to the Nmap Scripting Engine (NSE), Nmap had very limited vulnerability scanning capabilities.

However, with the addition of NSE scripts, it can now seek out (and in some cases exploit) vulnerabilities. This increases the flexibility and extensibility of Nmap by allowing users to write their own scripts with just a little bit of LUA scripting knowledge.

At the time of this writing, there are 433 NSE scripts provided as part of the Nmap download. To view the scripts that are available, list the directories that contain the NSE scripts.

On Windows hosts the scripts are typically in the following location:

```
c:\Program Files (x86)\Nmap\scripts\
```

On Linux hosts the scripts are typically in the following location:

```
/usr/local/share/nmap/scripts/
```

You can also check this reference for a listing and description: <http://nmap.org/nsedoc/>.

Since these are interpreted (and not compiled) scripts, it is possible to view the source to better understand what they do and even tweak them to behave differently. It is also quite helpful to start from one of these functioning scripts if LUA is not your forte. To activate Nmap NSE scripts, use the `--script` option and supply comma separated script names, categories, or directories:

```
--script filename|category|directory|expression[,...]
```

Since NSE was designed for a variety purposes, including extended version scanning, vulnerability detection and even exploitation, all scripts have one or more categories defined. This not only adds organization, but it can simplify the choices when running multiple scripts by being able to select a family of scripts based on their categories.

Category options are: auth, broadcast, brute, default, discovery, dos, exploit, external, fuzzer, intrusive, malware, safe, version, and vuln.

In addition to running scripts based on category, there are a couple really nice shortcuts, but they may not be the best choice in all circumstances:

`-sC` = Script Scanning: Equivalent to `--script=default` and will execute scripts which have the default category defined

`-A` = Aggressive scanning: Performs the following: Operating System (`-O`), Version (`-sV`), Script scanning (`-sC`) and traceroute (`--traceroute`).

The reason why caution should be applied when running either of these options is that default

scripts can include scripts in the intrusive category, which have the potential to crash systems or make printers go crazy.

Instead, it may be desirable to exercise more control over which NSE scripts are executed by using the boolean AND with the “safe” category. For example (Figure 5):

```
nmap -sS -T4 -Pn --script "default and safe" <host>
nmap -sS -T4 -Pn --script "http-* and safe" <host>
nmap -sS -T4 -Pn --script "smb-* and safe" <host>
nmap -sS -T4 -Pn --script "safe" <host>
```

For more information, a great resource is Fyodor's NSE usage page: <http://nmap.org/book/nse-usage.html>.

Manipulating Output

As mentioned earlier, the output options for Nmap are normal (`.nmap`), greppable (`.gnmap`), and XML (`.xml`). We will now explore how we can manipulate these standard output formats to achieve even greater insight--which is particularly useful when viewing larger output files.

Carving Standard Output

The standard output files can be quite useful for generating statistics or feeding other tools. The `nmap` and `gnmap` files are good for pulling information such as:

- Unique open ports
- Number of unique open ports
- Open ports to feed into other tools
- Hosts that are listening on a particular port
- Number of hosts listening on a particular port

Unique open ports

```
grep " open " FullTCP.nmap | cut -f 1 -d ' ' | sort -nu
```

```
root@p0rkak:/scans# nmap -sS -T4 -Pn --script "smb-* and safe" 192.168.1.5
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-14 08:08 EDT
Nmap scan report for .home (192.168.1.5)
Host is up (0.833% latency).
Not shown: 993 filtered ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
554/tcp   open  rtsp
2809/tcp  open  iclap
5357/tcp  open  wldap
10243/tcp open  unknown
MAC Address: DC:85:DE: (Unknown)

Host script results:
_ smb-enum:
  ERROR: Failed to connect to browser service
_ smb-os-discovery:
  OS: Windows 8 9200 (Windows 8 6.2)
  NetBIOS computer name:
  Workgroup: WORKGROUP
  System time: 2013-05-14T08:08:36-04:00
_ smb-security-mode:
  Account that was used for smb scripts: <blank>
  User-level authentication:
  SMB Security: Challenge/response passwords supported
  Message signing disabled (dangerous, but default)

Nmap done: 1 IP address (1 host up) scanned in 38.79 seconds
```

Figure 5. Results of an Nmap script scan against a Windows 8 host


```
22/tcp
23/tcp
80/tcp
135/tcp
--snip--
```

Number of unique open ports

```
grep " open " FullTCP.nmap | cut -f 1 -d ' ' |
    sort -nu | wc -l
```

16

Feed open ports into another tool (space separated)

```
grep " open " FullTCP.nmap | cut -f 1 -d ' ' |
    sort -nu | cut -f 1 -d '/' | xargs
```

```
22 23 80 135 139 443 445 902 912 992 4445 4567
    5357 8080 8443 49155
```

Feed open ports into another tool (comma separated)

```
grep " open " FullTCP.nmap | cut -f 1 -d ' ' |
    sort -nu | cut -f 1 -d '/' | xargs | sed 's/ /,/g'
```

```
22,23,80,135,139,443,445,902,912,992,4445,4567,535
    7,8080,8443,49155
```

Hosts that are listening on a particular port (telnet used below as an example)

```
grep 23/open/tcp FullTCP.gnmap | cut -f 2 -d ' '
```

```
192.168.1.1
192.168.1.2
```

(Replace port 23 with whatever port number you wish to find other ports/hosts of interest)

How many hosts are listening on a particular port (telnet used below as an example)

```
grep 23/open/tcp FullTCP.gnmap | cut -f 2 -d ' ' |
    wc -l
```

2

(Replace port 23 with whatever port number you wish to find other ports/hosts of interest)

Convert to HTML

We just illustrated the usefulness of normal and greppable output files, however XML files are al-

so useful. Nmap XML files can be converted into HTML format for easy viewing within a web browser. This is possible through using Extensible Stylesheet Language Transformations (XSLT).

Detailed information on various options can be found here: <http://nmap.org/book/output-formats-output-to-html.html>.

One very reliable method is using the xsltproc tool (from <http://xmlsoft.org/XSLT/>). Using it is simple and the result is portable. Use the Nmap XML file for input and use -o to specify the output file.

Syntax is below

```
root@DVORAK:~/scans# xsltproc TopTCP.xml -o
    TopTCP.html
root@DVORAK:~/scans# firefox TopTCP.html
```

Result: see Figure 6.

Notice that the output is very organized and port information is even displayed in easy to read tables. Navigation is made convenient through the

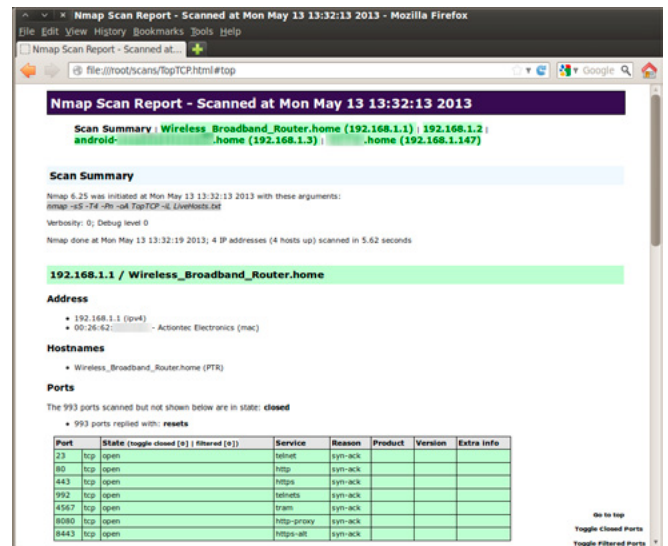


Figure 6. HTML result of using xsltproc

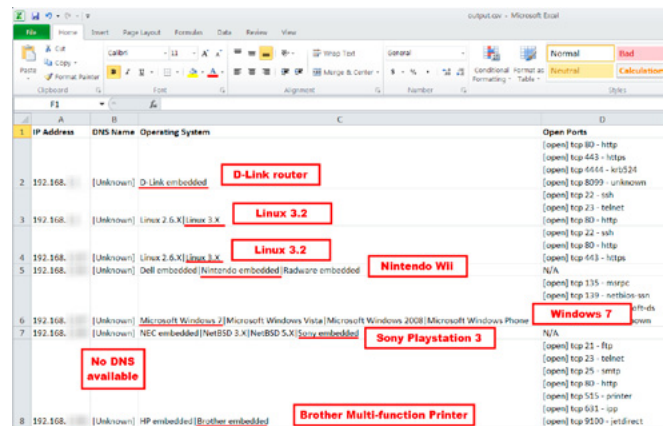


Figure 7. Output of the xml to CSV processing tool. Displays IP address, DNS name, operating system, and port information

use of the included hyperlinks and search functionality provided by the browser.

Processing Script

One thing that our customers always appreciate is an Excel spreadsheet of hosts, ports, services, operating systems, and other scan details. Excel is a natural fit for this type of data because it makes sorting and filtering a breeze. There are a few tools online that can help organize output from Nmap scans, including a free tool being released from FireEye Labs. This tool consumes an Nmap XML file and converts it to a CSV containing the following fields: IP address, DNS name, Operating System, and ports. Follow the FireEye blog (<http://www.fireeye.com/blog/>) for the latest information (Figure 7).

Summary

With little-to-no training, anyone can scan a network using Nmap by simply specifying the host or target range. While Fyodor has done a great job making Nmap default options as efficient as possible, you should still be able to improve scanning efficiency by customizing your scans. Hopefully, by sharing a proven scan methodology and tricks for manipulating the output, we have provided ideas

for you to refine your Nmap scan strategy and get the most from your efforts.

TONY LEE

Tony Lee has more than nine years of professional experience pursuing his passion in all areas of information security. He is currently a principal security consultant at FireEye Labs, in charge of advancing many of the network penetration testing service lines. His interests of late are Citrix and kiosk hacking, post exploitation tactics, and malware research. As an avid educator, Tony has instructed thousands of students at many venues worldwide, including government, universities, corporations, and conferences such as Black Hat. He takes every opportunity to share knowledge as a contributing author to Hacking Exposed 7, frequent blogger, and a lead instructor for a series of classes. He holds a Bachelor of Science degree in computer engineering from Virginia Polytechnic Institute and State University and a Master of Science degree in security informatics from The Johns Hopkins University.

Email: Tony.Lee-at-FireEye.com

Linked-in: <http://www.linkedin.com/in/tonylevt>

Special Thanks To

Bill Hau, Dan Dumond, Dennis Hanzlik, Jacob Martinson, Ron Nguyen, Rudolph Araujo

a d v e r t i s e m e n t

IT-Securityguard

Lets secure IT



Android Vulnerability Scan



Web Penetration testing



Secure hosting

contact: contact@it-securityguard.com

www.it-securityguard.com

Install and Configure

A Working Port Scan Attack Daemon PSAD on Ubuntu

A lot of tutorials deal with nmap scanning and OS fingerprinting especially from the attackers point of view. I would like to enlighten a quick and dirty approach to get a portscan detector up and running to add to your defense-in-depth. In this tutorial we will install the portscan attack detector daemon, or psad, for short.

PSAD is capable of automatically adding iptables rules in order to block all traffic to and from one or more portscanning IP addresses. There are not that many hands-on websites dealing with psad for a specific Linux distro, and the ones that do exist, miss some essential details to get things working. So I thought, why not write a quick recipe that quickly gives you both a working iptables script and a working port scan detector with an automatical response option (psad).

What you will learn is adding yet another defense ring against portscans, in this case psad. That enables you to react quickly, and when correctly set-up, can save valuable time.

What you should know is navigating the command prompt and installing software on your distro of choice. Although in the examples I use Ubuntu/Mint, any other distro might do fine with only minor adaptations. Another benefit would be knowing how to install Vmware and or VirtualBox. This way you can safely test without the need for a DMZ setup or more than one machine on your LAN. Furthermore, I assume you know how to connect to a ssh server.

If you know iptables, even better. I assume you have Ubuntu 12.04 or a later version installed and xtables-addons.

Although installing xtables-addons on Ubuntu is straightforward:

```
~ $ sudo apt-get install xtables-addons
```

Or if you you are a bit more adventurous you can install the latest xtables-addons from Sourceforge: <http://xtables-addons.sourceforge.net/>.

Before we can compile from source we have to make sure all dependencies have been met:

```
~ $ sudo apt-get build-dep xtables-addons
```

copy the downloaded file to /tmp and unpack the file in one go:

```
~ $ cp xtables-addons-2.2.tar.xz /tmp && tar xJf
xtables-addons-2.2.tar.xz
```

go into the directory of the unpacked archive:

```
~ $ cd /tmp/ xtables-addons-2.2
```

compile and install everything:

```
~ $ sudo ./configure && make && make installation
(Figure 1)
```

If you didn't face any errors you should load a module such as `xt_psd` in order to check whether everything has been properly installed

~ \$ sudo modprobe xt_psd (Figure 2)

Installing psad and fwsnort is just one apt-get away:

~ \$ sudo apt-get psad fwsnort

For the installation of psad and dependencies I also refer to the following website: <http://bodhizazen.net/Tutorials/psad>. Although I prefer to install packages from a official repository as much as possible, your mileage may vary. You can perfectly well use ufw or the iptables example from the above mentioned website although I used in this example the desk firewall example from the nixcraft website: <http://www.cyberciti.biz/faq/linux-detect-port-scan-attacks/>.

Don't forget to check your specific network card name:

~ \$ sudo ifconfig -a (Figure 3)

In my case if I would want to use the wireless network card I would enter wlan0 as being the interface name in the firewall script.

You could also use my desktop firewall script below, most important are the LOG lines near the end.

For obvious reasons, psad uses the input from the LOG entries.

In order to run the firewall (bash script) you will have to make it executable eg:

~ \$ sudo chmod +x </path/to/script>/script

then "calling" the script: /root/scripts/eth0.sh OR ./root/scripts/eth0.sh OR sh /root/scripts/eth0.sh should start the firewall. The bashscript looks for a file in the /root/scripts directory for the file blocked_fw in which you can add those pesky IP addresses you want to block in to oblivion. You noticed the line PUB_IF="eth0" in the script.

Don't forget to change eth0 into whatever your network interface card is named!!!

eg:wlan0 for a wireless connection

If you don't create the file and directory the script will trigger an error.

Of course you can alter everything to your liking.

On Ubuntu you should edit /etc/rc.local and append your newly created script directory and firewall script name.

~ \$ sudo Nano -w /etc/rc.local (Figure 4)

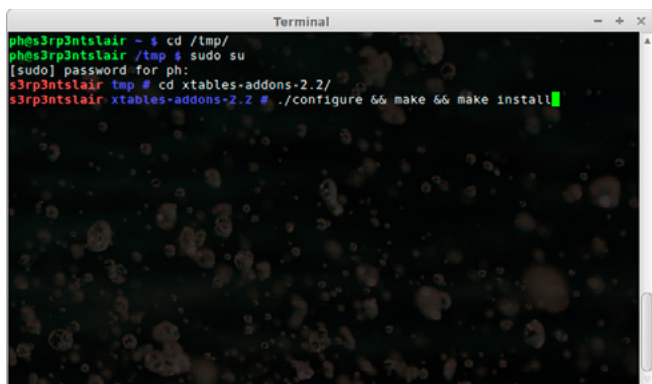


Figure 1. ~ \$ sudo ./configure && make && make installation

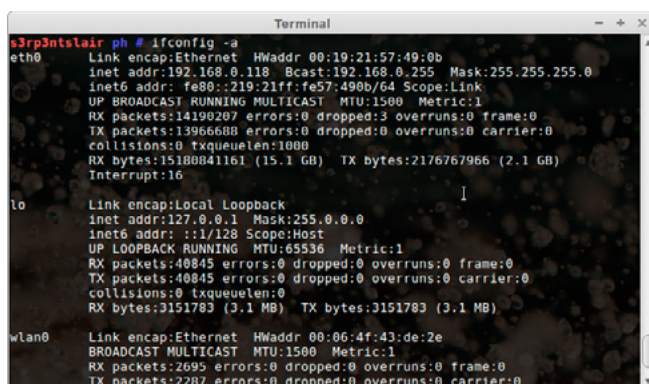


Figure 3. ~ \$ sudo ifconfig -a

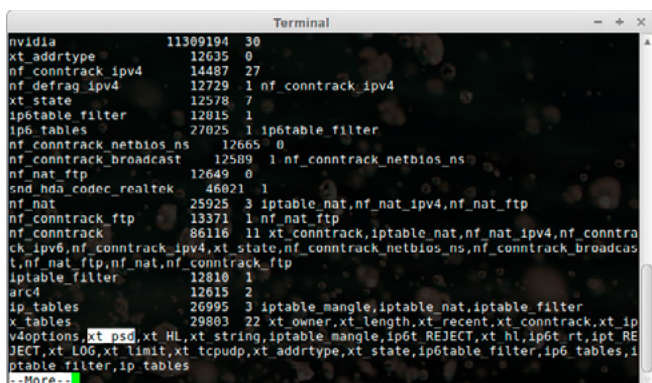


Figure 2. ~ \$ sudo modprobe xt_psd

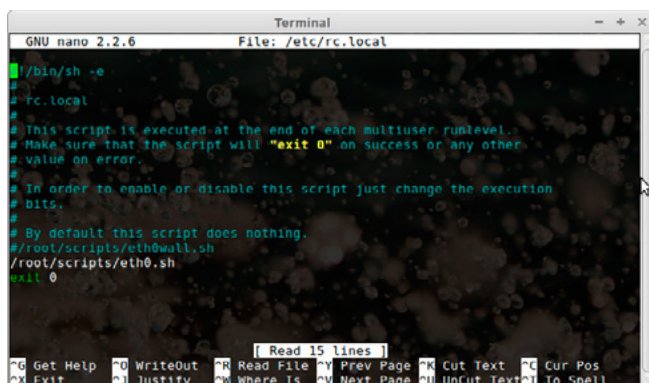


Figure 4. ~ \$ sudo Nano -w /etc/rc.local

Listing 1. Script (1)

```
#!/bin/bash
IPT="/sbin/iptables"
ipset=" /sbin/ipset"
SYSCTL="/sbin/sysctl"
echo "Starting IPv4 Wall..."
$IPT -F
$IPT -X
$IPT -t nat -F
$IPT -t nat -X
$IPT -t mangle -F
$IPT -t mangle -X
modprobe ip_conntrack
modprobe xt_string
modprobe ipt_string
modprobe ipt_ttl
modprobe ipt_TTL
modprobe xt_psd
modprobe ipt_psd
modprobe xt_ipv4options
modprobe ipt_ipv4options

BADIPS=$(egrep -v -E "^#|^$" /root/scripts/
        blocked.fw)
PUB_IF="eth0"

#$ipset -N spamhaus nethash
#unlimited
# Make certain attacks more difficult by edit-
        ing some system variables which
        may vary on some distros,
        #eg:execstack=..
echo "Setting sysctl IPv4 settings..."
$SYSCTL net.ipv4.ip_forward=0
$SYSCTL net.ipv4.conf.all.send_redirects=0
$SYSCTL net.ipv4.conf.default.send_redirects=0
$SYSCTL net.ipv4.conf.all.accept_source_route=0
$SYSCTL net.ipv4.conf.all.accept_redirects=0
$SYSCTL net.ipv4.conf.all.secure_redirects=0
$SYSCTL net.ipv4.conf.all.log_martians=1
$SYSCTL net.ipv4.conf.default.accept_source_
        route=0
$SYSCTL net.ipv4.conf.default.accept_redirects=0
$SYSCTL net.ipv4.conf.default.secure_redirects=0
$SYSCTL net.ipv4.icmp_echo_ignore_broadcasts=1
#$SYSCTL net.ipv4.icmp_ignore_bogus_error_mes-
        sages=1
$SYSCTL net.ipv4.tcp_syncookies=1
$SYSCTL net.ipv4.conf.all.rp_filter=1
$SYSCTL net.ipv4.conf.default.rp_filter=1
#$SYSCTL kernel.exec-shield=1
$SYSCTL kernel.randomize_va_space=2

$IPT -A INPUT -i lo -j ACCEPT
$IPT -A OUTPUT -o lo -j ACCEPT

# DROP all traffic
$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP

# block all bad ips in the file blocked.fw, the
        script loops through this file
for ip in $BADIPS
do
        $IPT -A INPUT -s $ip -j DROP
        $IPT -A OUTPUT -d $ip -j DROP
done
#create new chains for checking whether the
        traffic is valid, request from
        the wan shouldn't address #pri-
        vate address spaces for example
$IPT -N valid-src
$IPT -N valid-dst

$IPT -A INPUT -i ${PUB_IF} -j valid-src
$IPT -A FORWARD -i ${PUB_IF} -j valid-src
$IPT -A OUTPUT -o ${PUB_IF} -j valid-src
        -A FORWARD -o ${PUB_IF} -j valid-src

iptables -A INPUT -i ${PUB_IF} -j valid-src
iptables -A FORWARD -i ${PUB_IF} -j valid-src
iptables -A OUTPUT -o ${PUB_IF} -j valid-dst
iptables -A FORWARD -o ${PUB_IF} -j valid-dst
iptables -A valid-src -s 10.0.0.0/8 -j DROP
iptables -A valid-src -s 172.16.0.0/12 -j DROP
iptables -A valid-src -s 224.0.0.0/4 -j DROP
iptables -A valid-src -s 240.0.0.0/5 -j DROP
iptables -A valid-src -s 127.0.0.0/8 -j DROP
iptables -A valid-src -s 0.0.0.0/8 -j DROP
iptables -A valid-src -d 255.255.255.255 -j DROP
iptables -A valid-src -s 169.254.0.0/16 -j DROP
iptables -A valid-dst -d 224.0.0.0/4 -j DROP

#$IPT -A my_chain -m set --match-set spamhaus
        src,dst -j DROP

# sync
$IPT -A INPUT -i ${PUB_IF} -p tcp ! --syn -m
        conntrack --ctstate NEW -m
        limit --limit 5/m --limit-burst
        7 -j LOG --log-level 4 --log-
        prefix "Drop Syn"
$IPT -A INPUT -i ${PUB_IF} -p tcp ! --syn -m
        conntrack --ctstate NEW -j DROP

# Drop Fragments
```

```

#This is one line and ends with "Fragmented
Packets"
$IPT -A INPUT -i ${PUB_IF} -f -m limit --limit
5/m --limit-burst 7 -j LOG
--log-level 4 --log-prefix
"Fragmented Packets"
$IPT -A INPUT -i ${PUB_IF} -f -j DROP
# block bad stuff
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
ALL ALL -j DROP
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
ALL ALL -m conntrack --ctstate
NEW -j DROP
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
ALL FIN,URG,PSH -j DROP
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
ALL FIN,URG,PSH -m conntrack
--ctstate NEW -j DROP
#This is one line and ends with "NULL Packets"
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
ALL NONE -m limit --limit 5/m
--limit-burst 7 -j LOG --log-
level 4 --log-prefix "NULL Pack-
ets"
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
ALL NONE -m conntrack --ctstate
NEW -j DROP
#This one line and ends with -j DROP
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
SYN,RST SYN,RST -m conntrack
--ctstate NEW -j DROP
#This is one line and ends with "XMAS Packets"
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
SYN,FIN SYN,FIN -m limit
--limit 5/m -limit-burst 7 -j
LOG --log-level 4 --log-prefix
"XMAS Packets"
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
SYN,FIN SYN,FIN -m conntrack
--ctstate NEW -j DROP
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
FIN,ACK FIN -m limit --limit
5/m --limit-burst 7 -j LOG
--log-level 4 --log-prefix "Fin
Packets Scan"
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
FIN,ACK FIN -m conntrack
--ctstate NEW -j DROP # FIN
packet scans
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
ALL SYN,RST,ACK,FIN,URG -m
conntrack --ctstate NEW -j DROP
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
FIN,SYN,RST,ACK ACK -m conn-
track --ctstate NEW -m limit
--limit 1/sec -j LOG --log-pre-
fix "[Drop]ACK scan: "
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
FIN,SYN,RST,ACK ACK -m conn-
track --ctstate NEW -j DROP
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
ALL FIN,ACK -m conntrack
--ctstate NEW -m limit --limit
1/sec -j LOG --log-prefix
"[Drop]Maimon scan: "
$IPT -A INPUT -i ${PUB_IF} -p tcp --tcp-flags
ALL FIN,ACK -m conntrack
--ctstate NEW -j DROP
# Drop packets with bad tcp flags
$IPT -N BAD_FLAGS
$IPT -A INPUT -p tcp --tcp-option 64 -m recent
--set -j BAD_FLAGS
$IPT -A INPUT -p tcp --tcp-option 128 -m recent
--set -j BAD_FLAGS
# Drop packets that are too small Note:
$IPT -N SMALL
$IPT -A INPUT -p udp -m length --length 0:27 -m
recent --set -j SMALL
$IPT -A INPUT -p tcp -m length --length 0:39 -m
recent --set -j SMALL
$IPT -A INPUT -p icmp -m length --length 0:27 -m
recent --set -j SMALL
$IPT -A INPUT -p 30 -m length --length 0:31 -m
recent --set -j SMALL
$IPT -A INPUT -p 47 -m length --length 0:39 -m
recent --set -j SMALL
$IPT -A INPUT -p 50 -m length --length 0:49 -m
recent --set -j SMALL
$IPT -A INPUT -p 51 -m length --length 0:35 -m
recent --set -j SMALL
$IPT -A INPUT -m length --length 0:19 -m recent
--set -j SMALL
#Enforce SYN only connections on NEW connections
$IPT -A INPUT -p tcp ! --syn -m conntrack
--ctstate NEW -j LOG --log-pre-
fix "New not syn:"
$IPT -A INPUT -p tcp ! --syn -m conntrack
--ctstate NEW -j DROP
$IPT -A FORWARD -p tcp ! --syn -m conntrack
--ctstate NEW -j LOG --log-pre-
fix "New not syn:"

```



```

$IPT -A FORWARD -p tcp ! --syn -m conntrack
    --ctstate NEW -j DROP
#BLOCK OS Fingerprint Detection
$IPT -N os-fingerprint
$IPT -F os-fingerprint
$IPT -A os-fingerprint -p tcp --dport 0 -j DROP
$IPT -A os-fingerprint -p udp --dport 0 -j DROP
$IPT -A os-fingerprint -p tcp --sport 0 -j DROP
$IPT -A os-fingerprint -p udp --sport 0 -j DROP
$IPT -A os-fingerprint -p icmp --icmp-type
    address-mask-request -j DROP
$IPT -A os-fingerprint -p icmp --icmp-type
    address-mask-reply -j DROP
$IPT -N IPOPTS
$IPT -A INPUT -m ipv4options --flags
    lsrr,ssrr,router-alert -m
    recent --set -j IPOPTS
$IPT -A IPOPTS -j LOG --log-prefix "BAD IPOPTS
    SHUN " --log-tcp-sequence
    --log-tcp-options --log-ip-
    options -m limit --limit 1/
    second
$IPT -A IPOPTS -j DROP
$IPT -N PSD
$IPT -A INPUT -i eth+ -m psd -m limit --limit 5/
    minute -j PSD
$IPT -A INPUT -i wlan+ -m psd -m limit --limit
    5/minute -j PSD
$IPT -A INPUT -i ipsec+ -m psd -m limit --limit
    5/minute -j PSD
$IPT -A INPUT -i tun+ -m psd -m limit --limit 5/
    minute -j PSD
$IPT -A FORWARD -i eth+ -m psd -m limit --limit
    5/minute -j PSD
$IPT -A FORWARD -i wlan+ -m psd -m limit --limit
    5/minute -j PSD
$IPT -A FORWARD -i ipsec+ -m psd -m limit
    --limit 5/minute -j PSD
$IPT -A FORWARD -i tun+ -m psd -m limit --limit
    5/minute -j PSD
$IPT -A PSD -m limit --limit 1/second -j LOG
    --log-level info --log-prefix
    "PSD -- DROP " --log-tcp-
    sequence --log-tcp-options
    --log-ip-options
$IPT -A PSD -j DROP
# Allow full outgoing connection but no incom-
    ming stuff
$IPT -A INPUT -i eth0 -m conntrack --ctstate
    ESTABLISHED,RELATED -j ACCEPT
$IPT -A OUTPUT -o eth0 -m owner --uid-owner
    0 -m conntrack --ctstate
    NEW,ESTABLISHED,RELATED -j
    ACCEPT
$IPT -A OUTPUT -o eth0 -m owner --uid-owner
    2 -m conntrack --ctstate
    NEW,ESTABLISHED,RELATED -j
    ACCEPT
$IPT -A OUTPUT -o eth0 -m owner --uid-owner
    8 -m conntrack --ctstate
    NEW,ESTABLISHED,RELATED -j
    ACCEPT
$IPT -A OUTPUT -o eth0 -m owner --uid-owner
    33 -m conntrack --ctstate
    NEW,ESTABLISHED,RELATED -j
    ACCEPT
$IPT -A OUTPUT -o eth0 -m owner --uid-owner
    1000 -m conntrack --ctstate
    NEW,ESTABLISHED,RELATED -j
    ACCEPT
$IPT -A OUTPUT -o eth0 -m owner --uid-owner
    125 -m conntrack --ctstate
    NEW,ESTABLISHED,RELATED -j
    ACCEPT
$IPT -A OUTPUT -o eth0 -m owner --uid-owner
    999 -m conntrack --ctstate
    NEW,ESTABLISHED,RELATED -j
    ACCEPT
$IPT -A OUTPUT -o eth0 -m limit --limit 1/sec
    -j LOG --log-prefix "Out pkt
    dropped: "
#$IPT -A OUTPUT -o eth0 -m conntrack --ctstate
    NEW,ESTABLISHED,RELATED -j
    ACCEPT
#if you can't connect to the internet you prob-
    ably have to comment (place a
    hashtag # in front of)
#all the lines that have a line " m owner -uid-
    owner" in it and just use the
    two lines:
#$IPT -A INPUT -i eth0 -m conntrack --ctstate
    ESTABLISHED,RELATED -j ACCEPT
#$IPT -A OUTPUT -o eth0 -m conntrack --ctstate
    NEW,ESTABLISHED,RELATED -j
    ACCEPT
# allow ssh only (That is if you need and or use
    ssh, just uncomment what you
    will use)
#$IPT -A INPUT -p tcp --destination-port
    22 -m conntrack --ctstate
    NEW,ESTABLISHED,RELATED -j
    ACCEPT
#$IPT -A OUTPUT -p tcp --sport 22 -m
    conntrack --ctstate
    NEW,ESTABLISHED,RELATED -j
    ACCEPT

```

```
# allow incoming ICMP ping pong stuff
$IPT -A INPUT -p icmp --icmp-type 8 -m
    conntrack --ctstate
        NEW,ESTABLISHED,RELATED -j
        ACCEPT
$IPT -A OUTPUT -p icmp --icmp-type 0 -m
    conntrack --ctstate
        ESTABLISHED,RELATED -j ACCEPT
# No smb/windows sharing packets - too much
logging
$IPT -A INPUT -p tcp -i eth0 --dport 137:139
    -j DROP
$IPT -A INPUT -p udp -i eth0 --dport 137:139
    -j DROP
# Log everything else
# *** Required for psad ****
$IPT -A INPUT -j LOG
$IPT -A FORWARD -j LOG
$IPT -A INPUT -j DROP
# Start ipv6 firewall
# echo "Starting IPv6 Wall..."
#/root/scripts/start6.fw
#/root/scripts/spameater.sh
exit 0
```

Listing 2. Script (2)

```
~ $ sudo psad -H ### which displays the psad
    BLOCK chains

Chain PSAD_BLOCK_INPUT (1 references)
pkts bytes target prot opt in out source des-
tination
 0 0 DROP all -- * * 62.58.48.30 0.0.0.0/0
 0 0 DROP all -- * * 111.1.76.214 0.0.0.0/0
 0 0 DROP all -- * * 174.36.4.18 0.0.0.0/0

Chain PSAD_BLOCK_OUTPUT (1 references)
pkts bytes target prot opt in out source des-
tination
 0 0 DROP all -- * * 0.0.0.0/0 62.58.48.30
 0 0 DROP all -- * * 0.0.0.0/0 111.1.76.214
 0 0 DROP all -- * * 0.0.0.0/0 174.36.4.18

Chain PSAD_BLOCK_FORWARD (1 references)
pkts bytes target prot opt in out source des-
tination
 0 0 DROP all -- * * 0.0.0.0/0 62.58.48.30
 0 0 DROP all -- * * 62.58.48.30 0.0.0.0/0
 0 0 DROP all -- * * 0.0.0.0/0 111.1.76.214
 0 0 DROP all -- * * 111.1.76.214 0.0.0.0/0
 0 0 DROP all -- * * 0.0.0.0/0 174.36.4.18
 0 0 DROP all -- * *174.36.4.18 0.0.0.0/0
```

I named the firewall script `eth0wall.sh`. You can obviously name it whatever you want (Listing 1).

You can also view the firewall script at: <http://pastebin.com/PdZdwizu>.

Configure psad

~ \$ sudo nano -w /etc/psad/psad.conf ###de rest of the default settings will do fine

```
HOME_NET NOT_USED;
EXTERNAL_NET any;
ENABLE_SNORT_SIG_STRICT N;
ENABLE_AUTO_IDS Y;
ENABLE_AUTO_IDS_REGEX N; ### if you say Y here
                                psad will not auto-block
IPTABLES_BLOCK_METHOD Y;
TCPWRAPPERS_BLOCK_METHOD N;
AUTO_IDS_DANGER_LEVEL 2;
```

Make sure your router is whitelisted:

```
sudo nano -w /etc/psad/auto_dl.conf
```

and append:

```
eg: 192.192.178.1.1 0; ### all is well explained
```

in the file though, zero means:ignore

Remove rsyslog,install sysklogd,run the iptables script,restart psad

```
sudo apt-get install sysklogd
```

```
append to /etc/syslog.conf: kern.info |/var/lib/
                                psad/psadfifo
```

```
~ $ chmod +x desk.fw
```

```
~ $ sudo ./desk.fw
```

```
~ $ sudo psad --sig-update && sudo psad -H
```

```
~ $ sudo /etc/init.d/sysklogd restart
```

```
Terminal [redacted]@ubuntu:/home/[redacted]
File Edit View Terminal Go Help
TCP, Chain: INPUT, Count: 1, DP: 20034, SYN, Sid: 100029
Signature match: "BACKDOOR netbus Connection Ctttempt"
TCP, Chain: INPUT, Count: 1, DP: 12345, SYN, Sid: 100028
Signature match: "BACKDOOR Subseven connection attempt"
TCP, Chain: INPUT, Count: 1, DP: 27374, SYN, Sid: 100207
DST: 62.74.69.229

SRC: 85.190.0.3, DL: 2, Dsts: 2, Pkts: 38, Unique sigs: 2, Email alert:
Source OS fingerprint:
Linux (2.4.x kernel)

DST: 192.168.1.6, Local IP
Scanned ports: TCP 80-8080, Pkts: 38, Chain: INPUT, Intf: eth0
Signature match: "MISC HP Web JetAdmin communication attempt"
TCP, Chain: INPUT, Count: 4, DP: 8000, SYN, Sid: 100084
Signature match: "BACKDOOR DoomJuice file upload attempt"
TCP, Chain: INPUT, Count: 4, DP: 3128, SYN, Sid: 2375
DST: 85.190.0.3

Total scan sources: 3
Total scan destinations: 5

[+] These results are available in: /var/log/psad/status.out
```

Figure 5. ~ \$ sudo psad -H ### which displays the psad BLOCK chains

One way to test your psad installation is scanning your PC from a site such as nmap online.

That is if your server or pc is in a DMZ zone and thus directly accesible from the internet.

After a while online, without directly provoking a psad reaction yourself, you could see something like this: Listing 2 and Figure 5.

```
$host 85.190.0.3
$3.0.190.85.in-addr.arpa domain name pointer
proxyscan.freenode.net
```

Another way to test whether your newly installed configuration of psad is indeed blocking portscans, is scanning one PC from another one.

If you have an already implemented virtual lab with either VirtualBox or Vmware you should additionally scan from another (virtual) PC in order to check if your portscan detector works to your personal satisfaction, or in general.

In order to check whether psad works as it should, blocking every portscan attempt in its tracks, I use the following network layout. I login to a ssh

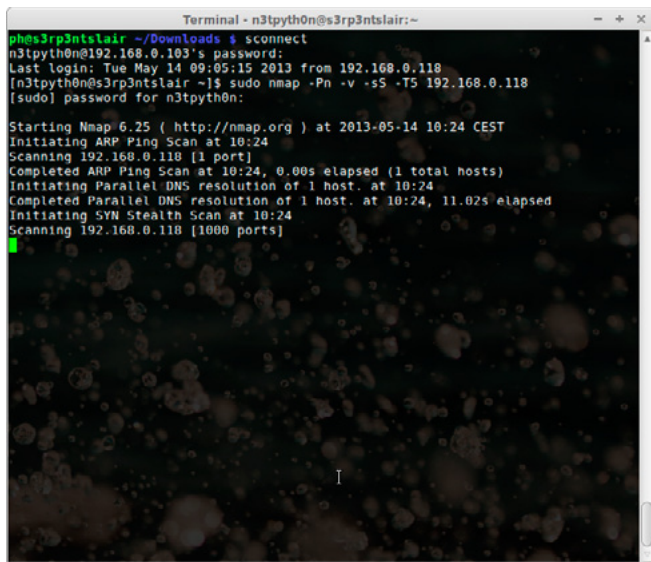


Figure 6. ~ \$ sudo nmap -Pn -v -sS -T5 192.168.0.118

server with IP 192.168.0.103 and launch a simple portscan with nmap against IP 192.168.0.118 (Listing 3).

In this setup, iwith psad installed on wks01 and scanning with nmap from server2: (Figure 6)

```
~ $ sudo nmap -Pn -v sS -T5 192.168.0.118
```

This scan should trigger a response from psad in which it writes automatically iptables rules in order to block all traffic to and from the IP address from which you just scanned.

To check if indeed the attacker's IP address (in this case 192.168.0.103) has been added to the auto-block chains you should enter the following:

```
~ $ sudo psad -S
```

You will notice the psad daemon has automatically added the IP 192.168.0.103 for auto-block Furthermore, the attackers IP has been blocked for a time period of 3600 seconds (Figure 7). You can change the time period to your liking in /etc/psad/psad.conf.

Or a bit easier for the eyes

```
~ $ sudo psad -L
```

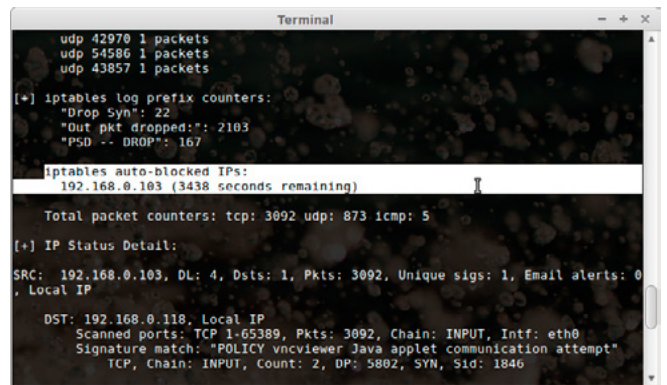
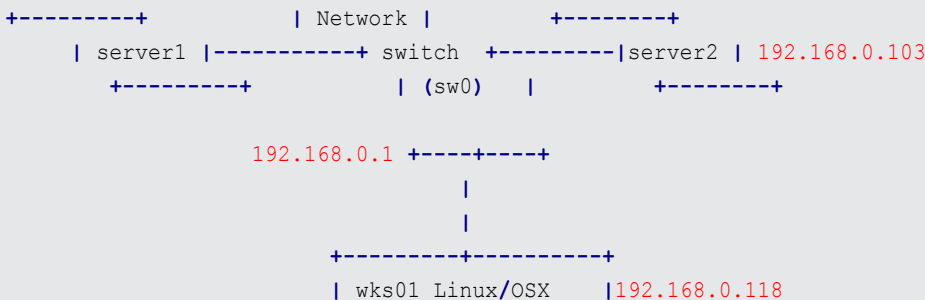


Figure 7. The Attackers IP Has Been Blocked for a Time Period of 3600 Seconds

Listing 3. Network layout



This will show you all the DROP chains that have been automatically created. (Figure 8)

To flush the psad rules added to iptables because you blocked an IP you need by accident or psad.config isn't fully configured you can always issue: psad -F Or "sudo psad -F" (without quotes). Your test attackers IP is released from the DROP chains and you can experiment again.

Now let's see if we could use some decoys when scanning with nmap in order to hide the real attacking ip (192.168.0.103) among the decoys (Figure 4).

```

Terminal
phes3rp3ntslair ~ $ sudo psad -L
[+] Listing chains from IPT AUTO_CHAIN keywords...

Chain PSAD_BLOCK_INPUT (1 references)
pkts bytes target prot opt in out source destination
1248 61776 DROP all -- * * 192.168.0.103 0.0.0.0/0

Chain PSAD_BLOCK_OUTPUT (1 references)
pkts bytes target prot opt in out source destination
3 0 0 DROP all -- * * 0.0.0.0/0 192.168.0.103

Chain PSAD_BLOCK_FORWARD (1 references)
pkts bytes target prot opt in out source destination
3 0 0 DROP all -- * * 0.0.0.0/0 192.168.0.103
3 0 0 DROP all -- * * 192.168.0.103 0.0.0.0/0

phes3rp3ntslair ~ $
    
```

Figure 8. All the DROP Chains That Have Been Automatically Created

Use the following syntax: Listing 4 and Figure 9-11.

With psad it should be possible to arrange all an attacker sees when trying to guess the OS it sees something like:

```

All 1000 scanned ports on 192.168.0.118 are filtered
MAC Address: 00:19:21:57:49:0B (Elitegroup
Computer System Co.).
Too many fingerprints match this host to give
specific OS details
Network Distance: 1 hop
TRACEROUTE
    
```

```

Terminal
phes3rp3ntslair ~ $ cat ~/.bashrc
## get rid of command not found ##
alias cd.='cd ..'
#ssh to music station
alias sconnect='ssh nmappython@192.168.0.103'
# a quick way to get out of current directory ##
alias ..='cd ..'
alias ...='cd ../..'
alias ....='cd ../../..'
alias .....='cd ../../../..'
alias .4='cd ../../..'
alias .5='cd ../../../..'
alias nowdate='date +%d-%m-%Y'
# Stop after sending count ECHO_REQUEST packets #
alias ping='ping -c 5'
# Do not wait interval 1 second, go fast #
alias fastping='ping -c 100 -s.2'
alias ports='netstat -tuln'
# do not delete / or prompt if deleting more than 3 files at a time #
alias rm='rm -I --preserve-root'

# confirmation #
alias mv='mv -i'
alias cp='cp -i'
    
```

Figure 9. in ~/.bashrc

Listing 4. Script (3)

```

# nmap -n -Ddecoy-ip1,decoy-ip2,your-own-ip,decoy-ip3,decoy-ip4 remote-host-ip
# nmap -n -D192.168.1.5,10.5.1.2,172.1.2.4,3.4.2.1 192.168.1.5
    
```

What would happen **if** we entered the whole subnet **as** decoy?

```

~ $ sudo nmap -n -D192.168.1-254 192.168.0.118
    
```

perhaps simpler but equivalent to the following bash script

```

#!/usr/bin/bash
i=0
range=256
while [ $i -lt $range ]
do
nmap -n -D192.168.1.$i 192.168.0.118
    
```

~ \$ sudo nmap -n -D192.168.1.5,10.5.1.2,172.1.2.4,3.4.2.1 -v -O --osscan-guess 192.168.0.118 (If it works who knows, I would like to get some info about the OS running on the other side). You probably noticed I entered sconnect at the top. It **is** just an alias I wrote in ~/.bashrc . I don't like to type to much of the same in repetition. (Figure 9 and 10)

Alias syntax:

```
alias name='command'
```

```
alias name='command arg1 arg2'
```

```
HOP RTT ADDRESS
10.35 ms 192.168.0.118
OS and Service detection performed. Please report
any incorrect results at http://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in
37.00 seconds.
```

However, nmap isn't the only tool with which you can fingerprint the OS.

Ettercap is perhaps a lesser known tool, more often thought of as a sniffer only tool for (switched) LANs.

There are many OS detection techniques that can be used without sending a single packet from your host (that don't trigger an action from an IDS arrangement such as psad/fwsnort). Although not really reliable is banner grabbing.

Some of the tools most commonly used are p0f and ettercap. I'm going to discuss p0f and ettercap here and will show you an alternative to nmap fingerprinting. Although ettercap is great for MITM (man in the middle attacks) one less known feature is the fingerprinting of a remote OS. Ettercap can filter content on the fly and has many other interesting techniques up its sleeve we will not discuss here. Ettercap supports active and passive

```
Terminal - n3tpython@es3rp3ntslair:~
n3tpython@es3rp3ntslair ~$ sconnect
n3tpython@192.168.0.103's password:
Last login: Tue May 14 13:23:40 2013 from 192.168.0.118
[n3tpython@es3rp3ntslair ~]$ sudo nmap -n -D192.168.1.5,10.5.1.2,172.1.2.4,3.4.2.1 -v -O --osscan-guess 192.168.0.118
[sudo] password for n3tpython:
Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-14 13:26 CEST
Initiating ARP Ping Scan at 13:26
Scanning 192.168.0.118 [1 port]
Completed ARP Ping Scan at 13:26, 0.01s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 13:26
Scanning 192.168.0.118 [1000 ports]
```

Figure 10. in ~/.bashrc

dissection of many protocols (even ciphered ones) and includes a lot of features for network and host analysis. Let's start with installing ettercap.

```
~ $ sudo apt-get install ettercap
```

Once our tool is installed we are ready to fire it up. We are going to use the ncurses (CLI) interface.

```
~ $ sudo ettercap -C
```

Step 1: Open the menu "sniff", and select "Unified sniffing"

Step 2: Choose the network interface to use: eth0

```
Terminal
ph0s3rp3ntslair ~$ sudo psad -L
[+] Listing chains from IPT_AUTO_CHAIN keywords...

Chain PSAD_BLOCK_INPUT (1 references)
pkts bytes target prot opt in out source destination
1260 59872 DROP all -- * * 172.1.2.4 0.0.0.0/0
1274 60488 DROP all -- * * 192.168.1.5 0.0.0.0/0
1300 60000 DROP all -- * * 192.168.0.103 0.0.0.0/0
1300 61632 DROP all -- * * 3.4.2.1 0.0.0.0/0

Chain PSAD_BLOCK_OUTPUT (1 references)
pkts bytes target prot opt in out source destination
0 0 DROP all -- * * 0.0.0.0/0 172.1.2.4
0 0 DROP all -- * * 0.0.0.0/0 192.168.1.5
0 0 DROP all -- * * 0.0.0.0/0 192.168.0.10
0 0 DROP all -- * * 0.0.0.0/0 3.4.2.1

Chain PSAD_BLOCK_FORWARD (1 references)
pkts bytes target prot opt in out source destination
0 0 DROP all -- * * 0.0.0.0/0 172.1.2.4
0 0 DROP all -- * * 172.1.2.4 0.0.0.0/0
0 0 DROP all -- * * 0.0.0.0/0 192.168.1.5
0 0 DROP all -- * * 192.168.1.5 0.0.0.0/0
0 0 DROP all -- * * 0.0.0.0/0 192.168.0.10
0 0 DROP all -- * * 192.168.0.103 0.0.0.0/0
0 0 DROP all -- * * 0.0.0.0/0 3.4.2.1
0 0 DROP all -- * * 3.4.2.1 0.0.0.0/0

ph0s3rp3ntslair ~$
```

Figure 11. This Technique Only Hides Your Source Address but Remote IPS / IDS Always Record and Logs Scan

Listing 5. Explanation

I know it didn't quite work out because the ssh connection froze. Probably we got dropped again. Let's see what it looks like at the other side.

As we thought, **not** only the decoys (3.4.2.1, 172.1.2.4, 192.168.1.5) but also the real attacking ip got blocked. As I quote from: <http://www.cyberciti.biz/tips/nmap-hide-ipaddress-with-decoy-ideal-scan.html>

"This technique only hides your source address but remote IPS / IDS always record **and** logs scan" Which we see **is** true (Figure 10).

By now you know why it **is** useful to whitelist some ip-addresses in "/etc/psad/auto_d1", because when we would enter `sudo nmap -n -D192.168.0.1 -v -O --osscan-guess 192.168.0.118` it might be possible you shoot yourself **in** the foot **and** psad blocks your gateway **in** this case (well mine **in** this setup).

Step 3: Open the menu “Start”, and select “Start sniffing”

Step 4: Open the menu “View”, and select “profiles”. In the main window you will see: “Collected passive profiles”

Step 5: Start your favorite browser, and type the address of the server you want to sniff.

Right after you have launched your web browser if you switch to the terminal where ettercap is running you can see several addresses ettercap has caught. If you opened a profile you would see something as in (Figure 12).

`p0f` is most ideal when you work with it on a host that is getting connections.

“The tool can be operated in the foreground or as a daemon, and offers a simple real-time API for third-party components that wish to obtain additional information about the actors they are talking to.

Common uses for `p0f` include reconnaissance during penetration tests; routine network monitoring; detection of unauthorized network interconnects in corporate environments; providing signals for abuse-prevention tools; and miscellaneous forensics.” <http://lcamtuf.coredump.cx/p0f3/README>.

Notice the IP address (192.168.0.118) that made it somewhat more difficult to remotely guess the OS

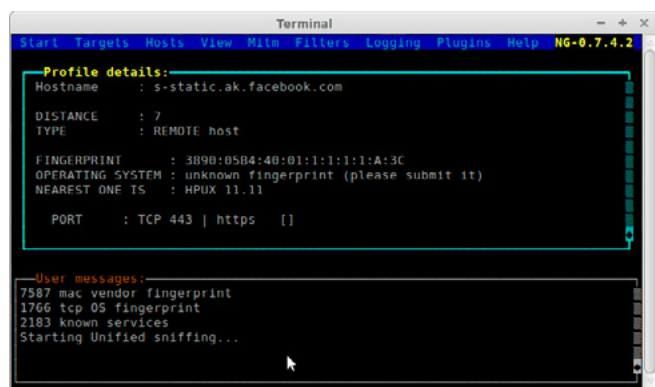


Figure 12. You Can See Several Addresses Ettercap Has Caught

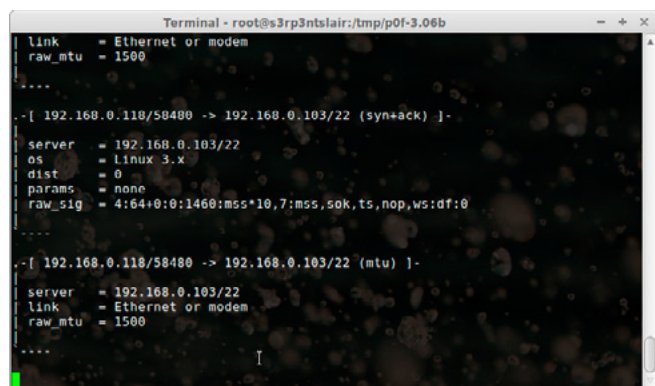


Figure 13. You Will See Almost Directly the Results

References

- <http://cipherdyne.org/psad/docs/>
- <http://bodhizazen.net/Tutorials/psad>
- <http://tjharmsen.blogspot.nl/2012/09/install-and-configure-working-port-scan.html>
- <http://www.cyberciti.biz/faq/linux-detect-port-scan-attacks/>
- <http://www.cyberciti.biz/tips/linux-scanning-network-for-open-ports.html>

Top 30 Nmap Command Examples For Sys/ Network Admins

- <http://www.cyberciti.biz/networking/nmap-command-examples-tutorials/>
- <http://lcamtuf.coredump.cx/p0f3/README>

from. However, the other way around when host (192.168.0.118) connects to host (192.168.0.103) who has an instance of `p0f` running (Figure 8) the OS details popup very fast, actually, right the moment `ssh user@192.168.0.103` is entered.

To test `p0f` enter on the host you are going to connect to: First install `p0f` if you haven't already

```
~ $ sudo apt-get install p0f
```

Running `p0f` is easy:

```
~ $ sudo p0f -i <network-interface>
```

While `p0f` is running on one PC try to connect to it from another PC via SSH.

You will see almost immediately the results. (Figure 13)

TO DO

PSAD enables you to execute a script upon a portscan detection directed at the IP address of the possible attacker. All you have to do besides writing a script, of course, is edit `/etc/psad/psad.conf`. You could write a script that tries to grab an abuse e-mail address and automatically sends the psad statistics attached to the abuse department of the network admin of the domain the attacker's IP address belongs to.

PETER HARMSSEN

I'm Peter Harmsen, MCSA. Autodidact and I like to travel through (network) security and OSS land. While beta-testing hakin9 and pentestmag articles I thought of writing an article myself. I hope the article is both fun and usefull.

Introduction to Nmap as a Computer Forensics Tool

Writing an article about Digital Forensics is always a challenge, and the reasons are multiple: the complexity of the argument, the level of technology involved, and the forensic approach itself.

There are a lot of tools and areas where digital forensics can be applied, and thousands of tools that can be used. But a challenge is always a good thing because it lets us focus and think clearer. So when I was proposed to write an article on Nmap and forensics, I accepted.

The first question I asked myself was: what can I write about, and I started to think about a lot of different approaches, then I realized that the best approach would be to start from the basics: what does digital forensics mean?

Then when I start to write other questions comes to my mind, and I have had to drive all those ideas toward the target this article was about Nmap and forensic, so I clearly picture in my mind a simple path that started what means Forensic in the digital world, then exploring a specific area, the network, at the end introducing a great tool like Nmap and finally exploring how to use Nmap for forensic use.

Since the argument is particularly wide and complex I choose an introductory approach, in order to be able to develop and dig deeper in future articles, hoping using also your feedback.

I hope you appreciate the approach and continue to read.

What Means Forensics (to me)

We all have had an introductory approach to forensic science in the past, we all saw thrillers and

maybe we have been fan of the CSI TV series. Analyzing a crime scene in order to find evidence that can be scientifically analyzed and used in a trial is not something unknown to us.

We learned in those movies and TV series that the basic of the forensic science is to apply science and technology in order to discover evidence of a law\policy infraction or crime, and beside the fact not always what we see on TV is real, this interpretation is correct. May be tools and procedures are not strictly real but gave us a basic understanding of what Forensic Science is.

Through the history [1] we've seen a great development of forensic instruments and branches: from pathology to entomology, from engineering to psychology several branches develop and we see new coming out as science and technology evolve.

No matter what branch or technique we're using the aim is always the same: to be able to analyze in detail a crime scene in order to find the truth using any physical available evidence.

Since this is a quite new science definition and rules are not well defined for any field, some areas are still in development, as for the computer and digital forensic, while other are in an advanced stage of development, as pathology for example. But still a definition would be useful so here some...

Definition: "Application of physical sciences to law in the search for truth in civil, criminal and so-

cial behavioral matters to the end that injustice shall not be done to any member of society”

Source: Handbook of Forensic Pathology College of American Pathologists 1990

And moving into the digital world:

Definition: “A methodical series of techniques and procedures for gathering evidence, from computing equipment and various storage devices and digital media, that can be presented in a court of law in a coherent and meaningful format”

Dr. H.B. Wolfe

Definition: “The preservation, identification, extraction, interpretation, and documentation of computer evidence, to include the rules of evidence, legal processes, integrity of evidence, factual reporting of the information found, and providing expert opinion in a court of law or other legal and/or administrative proceeding as to what was found.”

Steve Hailey, Cyber-security Institute

Computer Forensic is still a new branch of the Forensic world we have to face some aspects:

- Computer forensics is in its early or development stages
- It is different from other forensic sciences as digital evidence is examined
- There is a little theoretical knowledge based up on which empirical hypothesis testing is done
- Designations are not entirely professional
- There is a lack of proper training
- There is no standardization of tools
- It is still more of an “Art” than a “Science”

Working in the Cyber area require, nevertheless a quite standard approach, common to every forensic investigation. There are some “simple” rules that should be followed by any forensic investigator:

Identifying the crime

If there is no crime there is no need of a Forensic investigation, although it is always possible to use a forensic approach to proactively find possible areas where a crime can be pursued.

In any case, either if proactive or reactive, an investigation has to start with an object, a target, this is necessary in order to define instruments and tools that have to be used. Is quite different to suspect a database manipulation than a use of pornographic material in the work environment, and so different will be the tools used to gather evidences.

Gathering the evidence

This is the core of any forensic activity, evidences are what we will work on, but in Computer Forensic those evidences are partially physical and mostly digital. This require an extra effort in order to no pollute evidences due to our gathering.

The process should be non-destructive and should preserve the data collected and the environment we are working on.

There are several good reason to be cautious: usually we will work in working environment and the evidence could be stored on production systems. Stopping or harming those environments would be sometimes worse than the crime itself, so it is strongly suggested to not bring home the entire database and related servers of an e-commerce company making them stop the activity, would not be appreciated. Clearly we will have to find the best tradeoff between the collection of data and the preservation of the work environment.

Another good reason to be cautious is that digital data can be easily modified and this could invalidate our effort, trying to not change, modify the evidence is always a must, but here we need to be extra cautious.

We should also consider that sometimes law requirements force us to respect specific policy and rules, we can think of the privacy EU set of laws that makes really difficult to handle properly sensitive and personal data even during a forensic investigation.

If it is possible we should use as less as possible the original evidences, trying to use when possible duplicate.

Building a chain of custody

Forensic data usually need to be moved from the crime scene to a safer location where the forensic expert can analyze it. The chain of custody is fundamental when we’re moving digital data both if they are physical evidence, think of an HDD or USB keys, and digital as logs, scan reports or files. We have to be sure that nobody can tamper our evidences.

Analyzing the evidence

Finally we can analyze the evidence we gathered. Again most of the consideration of the previous points should be followed; we should never modify the original data, any modification should be reported in detail with any change of status, the analysis should be non-destructive when possible, we should use the appropriate tool and course of action considering the data we’re analyzing.

The 3 A's

1. Acquire evidence without modification or corruption
2. Authenticate that the recovered evidence is same as the originally seized data
3. Analyze data without any alterations

Presenting the evidence

Last but not least evidence should be presented in a clear and understandable manner, we have to consider:

- Who is listening to us, level of knowledge or expectation, and remember the golden kiss rule (keep it simple stupid).
- The objections and the evidences that can be used by the counterpart
- Obey rules of evidence, evidence should not be interpreted but should show solid facts.
- Never exceed the knowledge base, you should present just the evidences, any other consideration would be inappropriate.

Forensic and the network

Computer forensic evidences can be collected from a multitude of sources, data can be found in HDD's, USB keys, CD/DVD rom, Ram, Rom and Cmos memory and so on. We should also consider that data can be presented in raw form (think of a stream of byte in a drive), as files, as metadata, as process (sql queries for example) and so on. But the complex network environment can present other sources as protocols and the network itself.

Is, alas, out of the scope of this article to present all of the specific areas of investigation, as how to set up a proper investigation and build a case, this would require some books, so we will focus just about a specific aspect of a forensic investigation and a specific tools, this would be, of course, introductory as a complete dissertation on network forensic would exceed my space here, and my due date for the article.

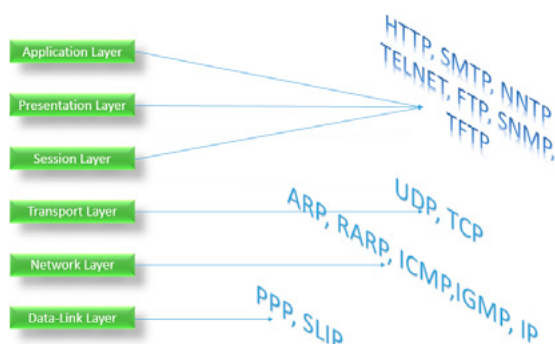


Figure 1. Network Protocol Overview with OSI layers correspondence

Se the Network can be itself analyzed for a forensic investigation, why we should do this and what can we could find?

The network is the communication backbone where our data flow, and thus can be used in a crime to modify, access, copy, tamper data and systems [3]. Being able to analyze this backbone is mandatory in any forensic investigation.

To be able to do so we should be able to understand network protocols and the various objects we can find in a network as:

- Computers
- Network devices as printer, scanner and so on
- Mobile computing devices as tablet, phones, laptop ...
- Network equipment's such as routers, switches, tappers, WAP (Wireless Access Point), firewalls, proxies, IDS ...
- ...

But we have also to relate them with network protocols at the various level, the operating systems used, application environment, authentication environment, management and security platforms and so on.

This is a quite complicated effort since, usually, there is not a complete and clear documentation nor a single source of information. We, as Forensic Investigator, need to create ourselves a map of the environment where we will perform our investigation.

Activities like listing all the active process on a machine or all the port currently used are commonly performed with tools or internal command of the various OS and are fundamental to understand and design the environment.

Network analysis is fundamental to trails the whole incident how the attack begins, which are the intermediate devices through which it pass and who was the victim. In order to obtain this goal we should collect evidences from log, firewalls, internetworking devices and files, and being able to make a network scan that can depict location and status of every device present on the network itself (Figure 1).

The first type of analysis is at the physical layer where we usually use:

- Sniffers, which put NICs in promiscuous mode that allow them to be used to collect digital evidence at the physical layer [4]
- SPANned ports, hardware taps help sniffing in a switched network

Sniffers collect traffic from the network and transport layers other than the physical and data-link layer.

Investigators should configure sniffers for the size of frames to be captured, the default size of frame that most sniffers capture by default is 68 bytes of Ethernet frame, It is advisable to configure sniffers to collect Ethernet frames of size 65535 bytes

The de facto standard of saving the gathered data from the network is in a tcpdump file with “*.dump” extension.

At the Data-Link Layer we usually check

- MAC address, a part of the data-link layer is associated with the hardware of a computer
- The ARP table of a router comes handy for investigating network attacks as the table contains IP addresses associated with the respective MAC addresses
- The DHCP database also provides as a means for determining the MAC addresses associated with the computer in custody The DHCP server maintains a list of recent queries along with the MAC address and IP Address

At the Network and transport layer we usually collect:

Authentication logs

- Shows accounts related to a particular event
- The IP address of the authenticated user gets stored in this log file

Application logs [5]

- Application logging is meant for the storage of auditing information, which includes information produced by application activity
- Web server logs help identify the system which was used as a means to commit the crime

Operating System logs

It maintains log of events such as errors, system reboot, shutdown, security policy changes, user and group management

But before enable logging one should bear in mind: what to log otherwise, it can result in over-collection of data making it difficult to trace the critical event.

Network device logs

Network devices such as router and firewalls are configured to send a copy of their logs to remote server as the memory for these devices is low.

The logs from network devices can be used as evidence for particular investigation on that network.

Some of those data requires administrative privileges to be collected, some other rely on tools able to help to gather read and analyze data. Since we will have to correlate data coming from different sources in order to reconstruct a process is mandatory to align all timestamps. So if this is not done we should collect all the time setting for each device providing a log and considerate eventual time gaps in order to normalize all the references.

Introduction to Nmap

There are some programs that a good network engineer should never forget, some of those tools are

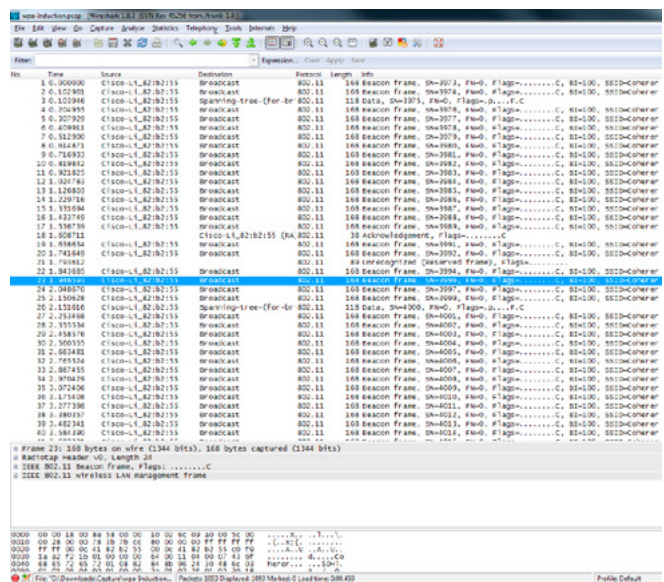


Figure 2. WireShark interface: a capture

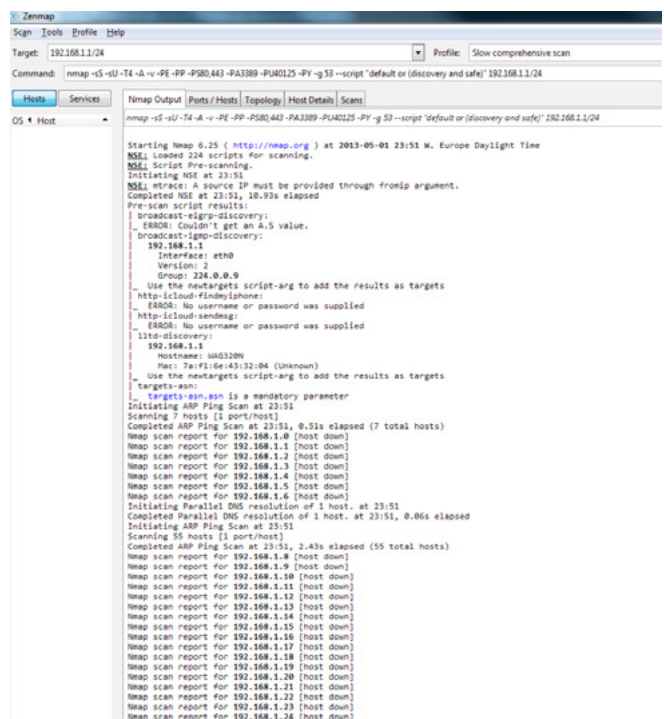


Figure 3. Nmap: Zenmap graphical interface

also useful in terms of forensic analysis and investigation (but also as hacking tools as well).

We all know Wireshark, a free great tool to read tcpdump files and perform network sniffing and packet analysis. Wireshark ecosystem provide plugin and add-on, for example a useful snort plugin, which amplify the software capability (Figure 2).

Retina is another great tool and respected security scanner.

Snort® is an open source network intrusion prevention and detection system (IDS/IPS).

Another great tool is Ettercap, the perfect companion for you man in the middle attacks but also a great too for network sniffing using arp poisoning.

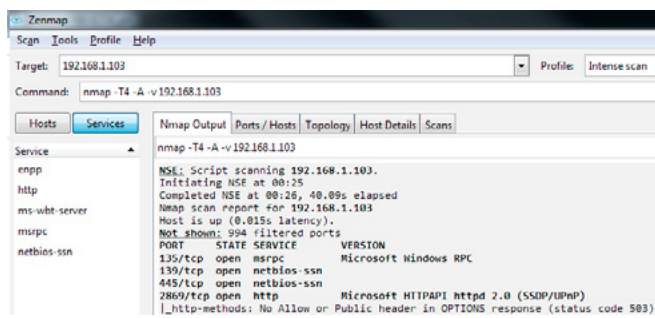


Figure 4. Zenmap, Host Scanning Output

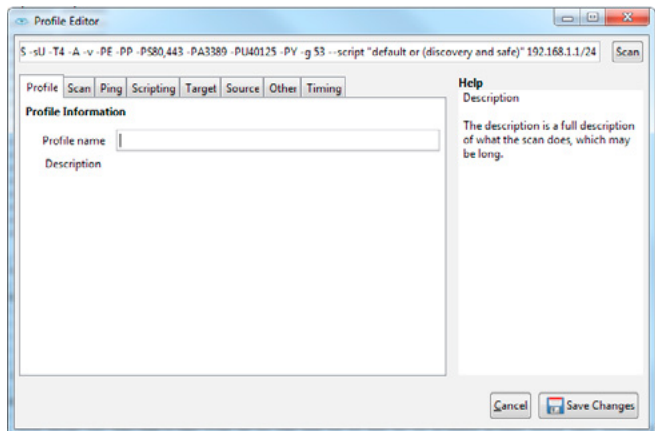


Figure 5. Zenmap Profile building interface

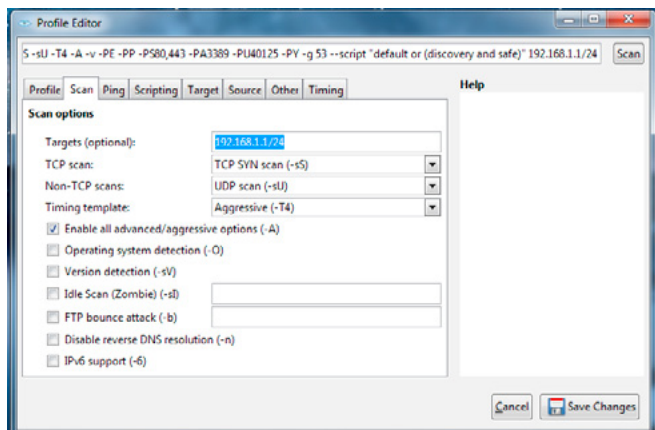


Figure 6. Zenmap profile options: scan

What I want to present you today is one of my favorite: Nmap (Figure 3).

The website description of Nmap is:

“Nmap (“Network Mapper”) is a free and open source (license) utility for network discovery and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. It was designed to rapidly scan large networks, but works fine against single hosts. Nmap runs on all major computer operating systems, and official binary packages are available for Linux, Windows, and Mac OS X. In addition to the classic command-line Nmap executable, the Nmap suite includes an advanced GUI and results viewer (Zenmap), a flexible data transfer, redirection, and debugging tool (Ncat), a utility for comparing scan results (Ndiff), and a packet generation and response analysis tool (Nping).

From <http://nmap.org/>

Nmap is a great tool for several reason, powerful and easy to use with its GUI (Zenmap) can be used in a more proficient way using its native scripting engine. There are dozens of ready-made scripts that allow you to discover network element and OS, vulnerability and so on.

The basic use of Nmap is for Network Scanning. What is Network Scanning?

Network scanning is the process of discovering active hosts on the network and information about the hosts, such as operating system, active ports, services, and applications.

Network scanning is comprised of the following four basic techniques:

- Network Mapping Sending messages to a host that will generate a response if the host is active
- Port Scanning Sending messages to a specified port to determine if it is active
- Service and Version Detection Sending specially crafted messages to active ports to generate responses that will indicate the type and version of service running
- OS Detection Sending specially crafted messages to an active host to generate certain responses that will indicate the type of operating system running on the host (Figure 4)

Listing 1. Example Script

```

Starting Nmap 6.25 ( http://nmap.org ) at
2013-05-02 00:25 W. Europe Daylight Time
NSE: LOADED 106 SCRIPTS FOR SCANNING.
NSE: SCRIPT PRE-SCANNING.
INITIATING ARP PING SCAN AT 00:25
SCANNING 192.168.1.103 [1 PORT]
COMPLETED ARP PING SCAN AT 00:25, 0.28S ELAPSED
(1 TOTAL HOSTS)
INITIATING PARALLEL DNS RESOLUTION OF 1 HOST. AT 00:25
COMPLETED PARALLEL DNS RESOLUTION OF 1 HOST. AT
00:25, 0.08S ELAPSED
INITIATING SYN STEALTH SCAN AT 00:25
SCANNING 192.168.1.103 [1000 PORTS]
DISCOVERED OPEN PORT 135/TCP ON 192.168.1.103
DISCOVERED OPEN PORT 445/TCP ON 192.168.1.103
DISCOVERED OPEN PORT 3389/TCP ON 192.168.1.103
DISCOVERED OPEN PORT 139/TCP ON 192.168.1.103
DISCOVERED OPEN PORT 2968/TCP ON 192.168.1.103
DISCOVERED OPEN PORT 2869/TCP ON 192.168.1.103
COMPLETED SYN STEALTH SCAN AT 00:25, 8.51S
ELAPSED (1000 TOTAL PORTS)
INITIATING SERVICE SCAN AT 00:25
SCANNING 6 SERVICES ON 192.168.1.103
COMPLETED SERVICE SCAN AT 00:25, 6.33S ELAPSED
(6 SERVICES ON 1 HOST)
INITIATING OS DETECTION (TRY #1) AGAINST 192.168.1.103
NSE: SCRIPT SCANNING 192.168.1.103.
INITIATING NSE AT 00:25
COMPLETED NSE AT 00:26, 40.09S ELAPSED
NMAP SCAN REPORT FOR 192.168.1.103
HOST IS UP (0.015S LATENCY).
NOT SHOWN: 994 FILTERED PORTS

PORT      STATE SERVICE      VERSION
135/TCP   OPEN  MSRPC        MICROSOFT WINDOWS RPC
139/TCP   OPEN  NETBIOS-SSN
445/TCP   OPEN  NETBIOS-SSN
2869/TCP  OPEN  HTTP         MICROSOFT HTTPAPI
          HTTPD 2.0 (SSDP/UPNP)
|_ HTTP-METHODS: NO ALLOW OR PUBLIC HEADER IN
OPTIONS RESPONSE (STATUS CODE503)
|_ HTTP-TITLE: SERVICE UNAVAILABLE
2968/TCP  OPEN  ENPP?
3389/TCP  OPEN  MS-WBT-SERVER MICROSOFT TERMINAL
          SERVICE
MAC ADDRESS: 00:19:D2:08:9B:A5 (INTEL)
WARNING: OSSCAN RESULTS MAY BE UNRELIABLE
BECAUSE WE COULD NOT FIND AT LEAST 1 OPEN AND 1
          CLOSED PORT
DEVICE TYPE: GENERAL PURPOSE|PHONE
RUNNING: MICROSOFT WINDOWS 7|VISTA|2008|PHONE

OS CPE: CPE:/O:MICROSOFT:WINDOWS_7::-:PROFESSIONAL
CPE:/O:MICROSOFT:WINDOWS_VISTA::-
CPE:/O:MICROSOFT:WINDOWS_VISTA::SP1
CPE:/O:MICROSOFT:WINDOWS_SERVER_2008::SP1
CPE:/O:MICROSOFT:WINDOWS
OS DETAILS: MICROSOFT WINDOWS 7 PROFESSIONAL,
MICROSOFT WINDOWS VISTA SP0 OR SP1, WINDOWS
SERVER 2008 SP1, OR WINDOWS 7, MICROSOFT WINDOWS
VISTA SP2, WINDOWS 7 SP1, OR WINDOWS SERVER
2008, MICROSOFT WINDOWS PHONE 7.5
UPTIME GUESS: 0.422 DAYS (SINCE WED MAY 01
          14:18:50 2013)
NETWORK DISTANCE: 1 HOP
TCP SEQUENCE PREDICTION: DIFFICULTY=258 (GOOD LUCK!)
IP ID SEQUENCE GENERATION: INCREMENTAL
SERVICE INFO: OS: WINDOWS; CPE: CPE:/
          O:MICROSOFT:WINDOWS

HOST SCRIPT RESULTS:
| NBSTAT:
| NETBIOS NAME: THINKPAD, NETBIOS USER:
<UNKNOWN>, NETBIOS MAC: 00:19:D2:08:9B:A5 (INTEL)
| NAMES
| HOME<00>          FLAGS: <GROUP><ACTIVE>
| THINKPAD<00>     FLAGS: <UNIQUE><ACTIVE>
| THINKPAD<20>     FLAGS: <UNIQUE><ACTIVE>
|_ HOME<1E>        FLAGS: <GROUP><ACTIVE>
| SMB-OS-DISCOVERY:
| OS: WINDOWS 8 PRO 9200 (WINDOWS 8 PRO 6.2)
| COMPUTER NAME: THINKPAD
| NETBIOS COMPUTER NAME: THINKPAD
| WORKGROUP: HOME
|_ SYSTEM TIME: 2013-05-02T00:26:03+02:00
| SMB-SECURITY-MODE:
| ACCOUNT THAT WAS USED FOR SMB SCRIPTS: GUEST
| USER-LEVEL AUTHENTICATION
| SMB SECURITY: CHALLENGE/RESPONSE PASSWORDS
          SUPPORTED
|_ MESSAGE SIGNING DISABLED (DANGEROUS, BUT DEFAULT)
|_ SMBV2-ENABLED: SERVER SUPPORTS SMBV2 PROTOCOL
TRACEROUTE
HOP RTT      ADDRESS
1 15.00 MS 192.168.1.103

NSE: SCRIPT POST-SCANNING.
READ DATA FILES FROM: C:\PROGRAM FILES (X86)\NMAP
OS AND SERVICE DETECTION PERFORMED. PLEASE
REPORT ANY INCORRECT RESULTS AT HTTP://NMAP.ORG/SUBMIT/ .
NMAP DONE: 1 IP ADDRESS (1 HOST UP) SCANNED IN
          58.98 SECONDS
RAW PACKETS SENT: 2038 (91.510KB) |
          RCVD: 40 (2.070KB)

```




Figure 7. Zenmap service button result: on the left all services detected

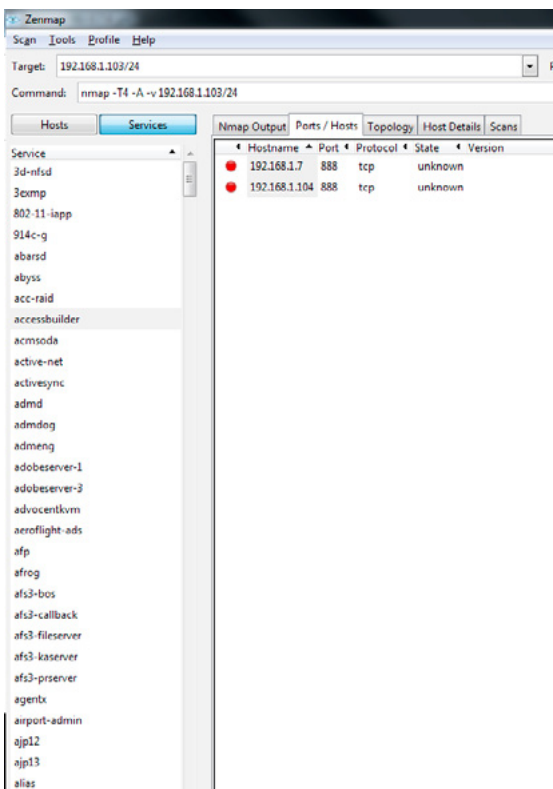


Figure 8. Host running a specific service and its status

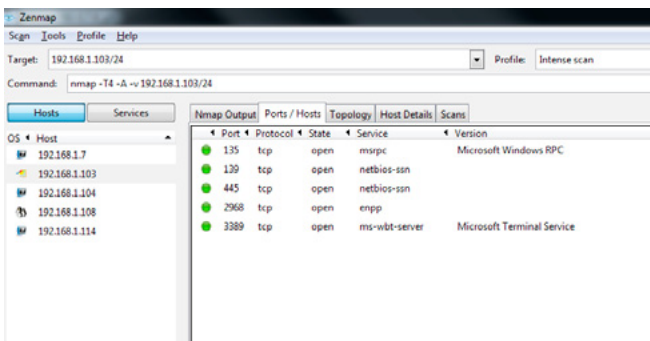


Figure 9. Host details

Just to understand what we're talking about let me show you an Nmap Output provided when scanning a single Host: I'm using the Windows version with Zenmap:

- Start zenmap\Nmap
- Then in the Target box write the Host you want to analyze. The next box, profile, allow you to choose a profile (kind of scan) to use
- Zenmap provides built-in profiles for the most common types of scans. This simplifies the scanning process by eliminating the need to manually specify a long string of arguments on the command line [6].
- Finally click the scan button and wait ...

This is the output I received: Figure 5 here different setting can be added as well as scripts and other parameters (Figure 6). See Listing 1.

As you can see Nmap performed a series of test trying to detect network ports open, the OS running, the services and so on.

We can of course scan not only a single Host but a network segment to find hosts in the neighborhood of host we need, for example if we put in the target field something like 192.168.1.103/24 we will obtain a more complex answer (Figure 7).

But with the Zenmap interface we can always easily focus on the services discovered during the scan that can be found on the left clicking on the Services button (Figure 8)

And so find which host is running that service and the relative status. On the other end if we want to put our focus on the hosts, we can simply click on Hosts and see what's cooking (Figure 9).

We can check also the topology (useful when routing or segmented environment are in place) (Figure 10)

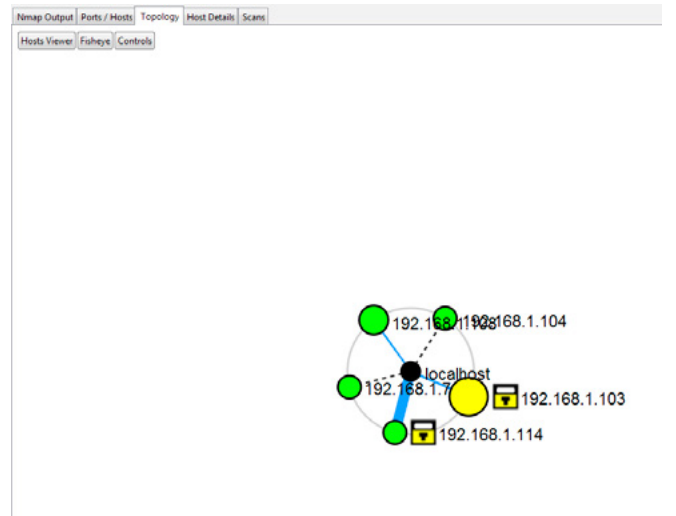


Figure 10. Zenmap Topology

And get the host results in a dedicated windows just clicking on Host Viewer (Figure 11).

As simple as at, Nmap can perform tremendous scans on our network and using the correct parameters can do quite everything. There are several references and books that cover all Nmap features so i will not spend time here explaining in detail all the option but i will name only those one can be particularly useful.

One great feature of Nmap is the capability to avoid firewall and IDS to be able to make our network discovery.

Firewalls and intrusion prevention systems are designed to prevent tools like Nmap from getting an accurate picture of the systems they are protecting but Nmap includes a number of features designed to circumvent these defenses.

Feature	Option
Fragment Packets	-f
Specify a Specific MTU	--mtu
Use a Decoy	-D
Idle Zombie Scan	-sI
Manually Specify a Source Port	--source-port
Append Random Data	--data-length
Randomize Target Scan Order	--randomize-hosts
Spoof MAC Address	--spooof-mac
Send Bad Checksums	--badsum

The idea is to bypass firewall and IDS\IPS policy in order to be able to fool the security device. For example changing the source port would allow us

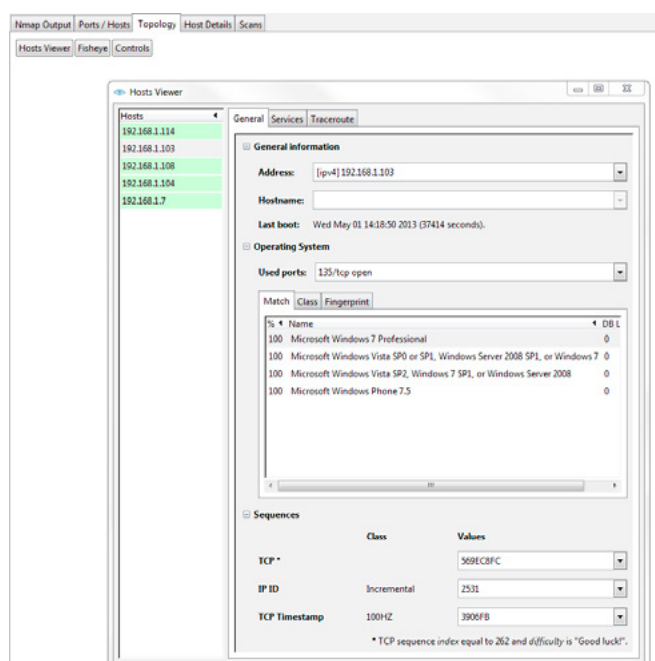


Figure 11. Zenmap Host Details from topology

to perform services tests bypassing an open fire-wall rule.

Nmap Scripting Engine Overview

The Nmap Scripting Engine (NSE) is a powerful tool that allows users to develop [7] custom scripts which can be used to harness Nmap's advanced scanning functions. In addition to the ability to write your own custom scripts, there are also a number of standard built-in scripts that offer some interesting features such as vulnerability detection and exploitation. A complete list of the built-in scripts for Nmap 5.00 can be found online at <http://www.nmap.org/nsedoc/>.

Feature	Option
Execute Individual Scripts	--script [script]
Execute Multiple Scripts	--script [script1,script2,etc]
Execute Scripts by Category	--script [category]
Execute Multiple Script Categories	--script [category1, category2]
Troubleshoot Scripts	--script-trace
Update the Script Database	--script-updatedb

Those parameters can be added in a custom profile to be able to use them without writing all the variables every time needed.

The scripts are also divided into category to make easier the use in specific profiles, the default are showed below:

Categories	Purpose
All	Runs all available NSE scripts
auth	Scripts related to authentication
default	Runs a basic set of default scripts
discovery	Attempts to discover in depth information about a target
external	Scripts that contact external sources (such as the whois database)
intrusive	Scripts which may be considered intrusive by the target system
malware	Scripts that check for open backdoors and malware
safe	Basic scripts that are not intrusive
vuln	Checks target for commonly exploited vulnerabilities

Using script categories is the easiest way to launch NSE built-in scripts – unless you know the specific script you want to run. Executing scripts by category, however, can take longer to complete since each category contains numerous scripts.

Nmap in the forensic environment

Basically when we are engaged for an investigation we should understand, at first, the very nature

References

- [1] Francis Galton (1822-1911)
- Made the first recorded study of fingerprints
 - Leone Lattes (1887-1954)
 - Discovered blood groupings (A,B,AB, & 0)
 - Calvin Goddard (1891-1955)
 - Allowed Firearms and bullet comparison for solving many pending court cases
 - Albert Osborn (1858-1946)
 - Developed essential features of document examination
- [2] Hans Gross (1847-1915)
- Made use of scientific study to head criminal investigations 1932
 - An FBI Lab was set up to provide forensic services to all field agents and other law authorities throughout the country 1984
 - FBI Computer Analysis and Response Team (CART) emerged 1991
 - International Law Enforcement meeting was conducted to discuss computer forensics & the need for standardized approach 1997
 - Scientific Working Group on Digital Evidence (SWGDE) was established to develop standards 2001
 - Digital Forensic Research Workshop (DFRWS) was held <http://www.dfrws.org/>
- [3] A modern computing systems can be accessed in a lot of different ways: physically, through usb devices, remotely through a physical network, remotely through a wifi, Bluetooth or GPRS/EDGE/UMTS/HSDPA network.
- [4] Usually a NIC (Network Interface Card) respond to packet sent only to its MAC Address (that is considered the Physical address for that network node). When in promiscuous mode a NIC process every packet even if the MAC address is not its own one.
- [5] Usually only administrator should has the privilege to access these log files
- [6] If the built-in scans don't meet your exact needs, you can create your own scan profile. To do this, simply access the profile editor by selecting Profile > New Profile from the Zenmap menu (or press CTRL + P on the keyboard)
- [7] Scripts for NSE are written in the Lua programming language. Unfortunately, programming in Lua is outside the scope of this article. For more information about Lua visit www.lua.org.
- [8] Nmap scan quick reference
Thursday 2 May 2013
14:34
nmap [Scan Type(s)] [Options] <host or net #1 ... [#N]>

Scan Options

- sT (TcpConnect)
- sS (SYN scan)
- sF (Fin Scan)
- sX (Xmas Scan)
- sN (Null Scan)
- sP (Ping Scan)
- sU (UDP scans)
- sO (Protocol Scan)
- sI (Idle Scan)
- sA (Ack Scan)
- sW (Window Scan)
- sR (RPC scan)
- sL (List/Dns Scan)

Ping detection

- P0 (don't ping)
- PT (TCP ping)

- PS (SYN ping)
- PI (ICMP ping)
- PB (= PT + PI)
- PP (ICMP timestamp)
- PM (ICMP netmask)

Output format

- oN(ormal) -oX(ml) -oG(rapeable) -oA(II)

Timing

- T Paranoid – serial scan & 300 sec wait
- T Sneaky – serialize scans & 15 sec wait
- T Polite – serialize scans & 0.4 sec wait
- T Normal – parallel scan
- T Aggressive – parallel scan & 300 sec timeout & 1.25 sec/probe
- T Insane – parallel scan & 75 sec timeout & 0.3 sec/probe
- host_timeout --max_rtt_timeout (default - 9000)
- min_rtt_timeout --initial_rtt_timeout (default - 6000)
- max_parallelism --scan_delay (between probes)
- resume (scan) --append_output
- iL <targets_filename> -p <port ranges>
- F (Fast scan mode) -D <decoy1 [,decoy2][,ME],>
- S <SRC_IP_Address> -e <interface>
- g <portnumber> --data_length <number>
- randomize_hosts -O (OS fingerprinting) -I (dent-scan)
- f (fragmentation) -v (verbose) -h (help)
- n (no reverse lookup) -R (do reverse lookup)
- r (dont randomize port scan) -b <ftp relay host> (FTP bounce)

[9] Ping Scan

Thursday 2 May 2013

15:10

```
# nmap -sP 192.168.2.0/24
```

```
Starting Nmap 4.50 (http://insecure.org) at
2007-12-28 11:40 EST
```

```
Host 192.168.2.1 appears to be up.
```

```
Host 192.168.2.3 appears to be up.
```

```
Host 192.168.2.4 appears to be up.
```

```
Nmap done: 256 IP addresses (3 hosts up) scanned
in 1.281 seconds
```

The ping scan sends an Internet Control Message Protocol (ICMP) echo request packet and a Transmission Control Protocol (TCP) acknowledge (ACK) packet to port 80 (TCP SYN packet if executed by an unprivileged user) to all specified targets and prints out information on the

of the crime or the incident. In the digital world this is not always possible since there can exist a mix of different situations, misconfigurations, errors and other things that can tamper the evidences.

We can be in presence of policy abuse, misuse of company resources, privacy law violations, information stealing but also malware, hacking, social engineering and so on. The spread of the digital violations is particularly wide so the first step is always the same, stand up and ask ourselves

why we're there... To understand what is happening and why they called us we need to interview our clients' customer to try to understand what is the complaint and what they have done as a reaction. Sometimes we can be lucky and we have to act upon an event that is still in place, think of a Ddos attack or a Web hacking, but most of the time we will be engaged when the "damage" is done.

Since this is usually a reactive action we should ask them to freeze as much as possible the situ-

hosts that responded. If the active target is on the same local Ethernet network, Nmap includes the Media Access Control (MAC) address and the associated manufacturer according to the Organizationally Unique Identifier (OUI). This is because Nmap uses the ARP scan (-PR) by default on the local Ethernet network. ARP scans are disabled by using the --send-ip command-line option. The ping scan does not scan ports or perform any other scan techniques. The ping scan can be used to create a network inventory, manage the asset database, and monitor system availability.

Nmap allows you to specify a variety of ICMP ping types. The ICMP type 8 echo request (-PE) expects an ICMP type 0 echo reply from an active host. The ICMP type 13 timestamp request (-PP) expects a type 14 timestamp reply from an active host and the ICMP type 17 address mask request (-PM) expects a type 18 address mask reply from an active host. Firewalls can be obstacles to ICMP discovery methods, as responses are often dropped. If you are scanning through a firewall, using one of the advanced host discovery techniques may offer better results.

Nmap offers several advanced methods to solicit replies from active hosts including the TCP SYN ping, TCP ACK ping, and the User Datagram Protocol (UDP) ping. The advanced methods can be combined (and also used with the -sP) or used individually.

The TCP SYN ping (-PS) creates a packet with the SYN flag set and sends it to specified ports on the target. By default, Nmap uses port 80, but you can specify a single port or multiple ports. If the specified port is closed the device will reply with an RST packet; if it is open it will reply with a SYN/ACK. Either response is acceptable for host discovery, because they both indicate that a device is active and responding. If no response is received, the target is either not active or the responses are being blocked by a firewall. The following shows the command-line options and output of a TCP SYN ping to port 80:

```
# nmap -sP -PS --reason 192.168.2.1-4
Starting Nmap 4.50 (http://insecure.org) at
2007-12-28 12:02 Eastern
StandardTime
Host 192.168.2.1 appears to be up, received syn-
ack.
Host 192.168.2.3 appears to be up, received
reset.
Nmap done: 4 IP addresses (2 hosts up) scanned
in 13.309 seconds
```

This example uses the --reason command-line option to show more detail on the response from the target hosts. In this case, we see that we received a SYN/ACK from 192.168.2.1 in response to our SYN packet, indicating that both the host and the port are active and

responding. Host 192.168.2.3 replied with a RST packet, indicating that the port is closed, but still revealing that the host is active and responding. The other two hosts in the target range, 192.168.2.2 and 192.168.2.4, did not respond, indicating that the host is either inactive or blocked by a firewall.

Nmap Port States

At first glance, you may think that a port can have two states: open and closed.

While this is true from the operating system's point of view, Nmap can detect other occurrences effecting state. Nmap detects the following six port states:

Open	Open ports have an active application accepting TCP connections or UDP packets.
Closed	Closed ports are accessible, but they do not have a listening application.
Filtered	Responses are blocked by a packet filter, therefore Nmap cannot determine if the port is open.

[10] Nmap Port States

At first glance, you may think that a port can have two states: open and closed.

While this is true from the operating system's point of view, Nmap can detect other occurrences effecting state. Nmap detects the following six port states:

Open	Open ports have an active application accepting TCP connections or UDP packets.
Closed	Closed ports are accessible, but they do not have a listening application.
Filtered	Responses are blocked by a packet filter, therefore Nmap cannot determine if the port is open.
Unfiltered	Unfiltered ports are accessible, but Nmap is unable to determine if they are open or closed. (ACK scan only)
Open filtered	Nmap is unable to determine if the port is open or filtered for scan types where open ports do not respond. (UDP, IP Proto, FIN, Null, Xmas scans)
Closed filtered	Nmap is unable to determine if a port is closed or filtered. (IP ID idle scan only)

ation preserving logs related to the timeframe we suspect the incident happened.

The first step of an investigation should always be a clear agreement between the customer and the investigator, law requirement and permission have to be evaluated and it is mandatory to have, always, a written authorization to perform data gathering since there could be sensible data involved.

Then usually is recommended to make an investigation Plan, this plan have to be a living corpus of action and can be modified upon the actual evidence collected, but it is important to always report any change with a reason, this could require the request for additional permission and authorization, do not make any move before have the written statement that authorizer you.

The third step should be freeze the crime scene, alas this is not always possible since we can be in presence of a working production environment, so we need to create the best picture possible to describe the boundaries. This is the moment for the log collection and the Nmap analysis.

Although good administrator always document their network, any external modification could change the very structure of the network without the administrator even know it, working side by side with the various system administrators and, if present, cyber security unit will ease the effort, but this will not avoid the need of a good network scan.

Network Mapping is therefore a mandatory operation, the aim is to detect and identify all the network nodes and all the running services.

Usually is a good course of action to make several map at different times and then compare the results with a tool like Ndiff.

The reason is that we need to be able to identify which variations occur to the network and understand if are legitimate or not.

This is a key point in the information gathering process since it eventually allow us to define specific targets that need further investigation, limiting the spread of the analysis.

While we perform the Nmap scan [7] we should also use Wireshark in order to capture all the traffic detect during the transaction. The reason for this precaution is that that way we can check at packet level the transactions to eventually understand anomalies. (Of course if you like more you can use tcpdump but Wireshark is a powerful and really easy tool that could save us hours of analysis).

One of the problem here is what level of scan we can perform on the network, if we use a simple ping scan [9], for example, we can trace only the node that are allowed to respond to igmp requests, this is not really useful in presence of firewall or

other security devices that block standard igmp packet. On the other end although incomplete this is a very light and fast scan that can help us to enumerate most of the device on the network.

OS discovery and port scanning are usually a normal companion of the basic Nmap scan test you should perform on the network, the result will outline which node are active and the status of the port [10] as well as the best guessing for the OS.

Running Nmap more heavy tests slow the scan and can provide performance problems to the network itself, so a vulnerability scan is useful but with some precaution. And again let me stress out that we need to require a written permission before performing any type of scan. This requirement is stricter if we consider wireless environment where also guest and external devices can be contacted or connected.

I will leave to Nmap documentation or future article a deeper dig on the various scan we can perform as: ping, port, OS, services and the more advanced available with the scripting engine as the Enhanced Network Discovery that perform whois lookups, perform additional protocol queries, and act as a client for the listening service to collect information such as available network shares.

Enhanced Version Detection	Perform complex version probes and attempt service brute-force cracking.
Vulnerability Detection	Execute probes to check for specific vulnerabilities.
Malware Detection	Execute probes to discover Trojan and worm backdoors.
Vulnerability Exploitation	Execute scripts to exploit a detected vulnerability.

Once the analysis has been performed we should be able to gather all the info and correlate all data together but this is something we we'll approach in the future.

Hope you enjoyed the ride.

ANTONIO IERANO



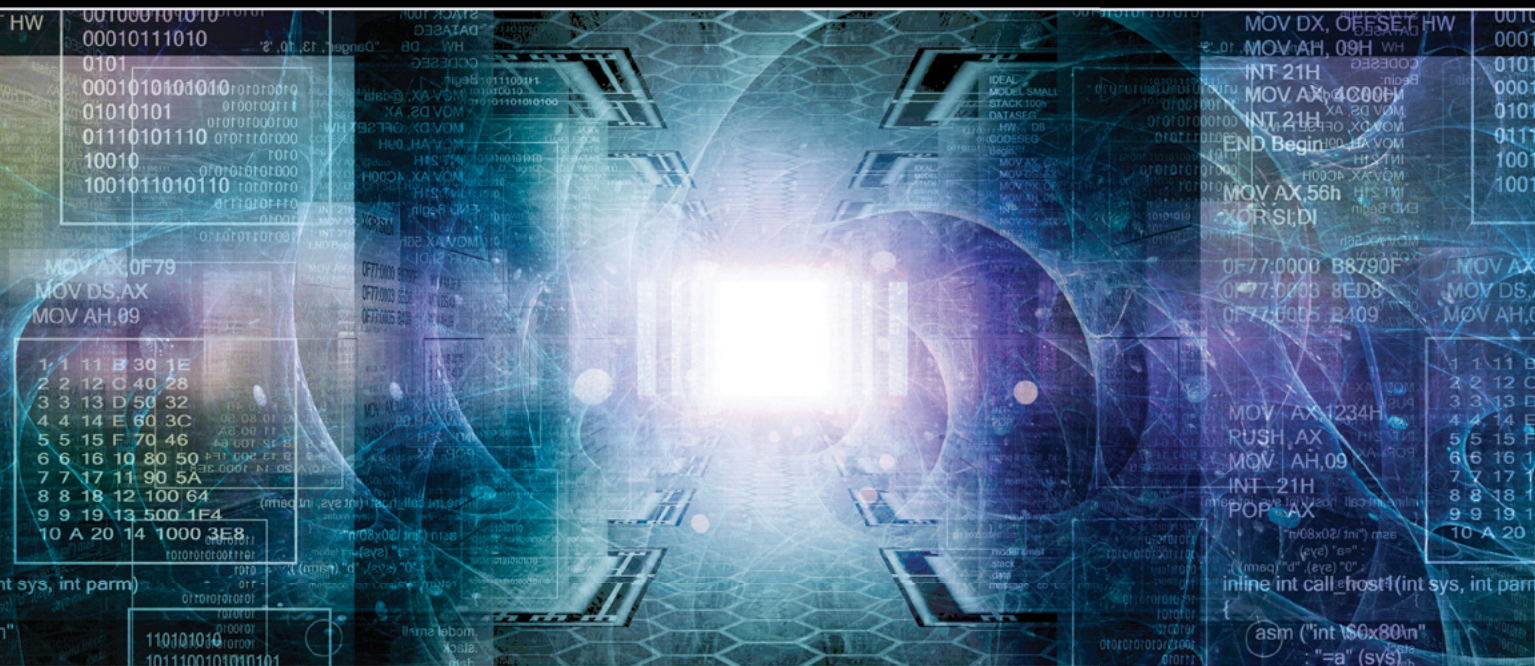
Antonio Ierano is an IT professional, marketing specialist, and tech evangelist with over 16 years of experience serving as a community liaison, subject matter expert, and high-profile trainer for key technologies and solutions. Mr

Ierano's experience includes acting as the public face of Cisco security technologies; leading pan-European technical teams in development of new Cisco security products; and serving as a key public speaker and trainer on behalf of new high-tech products. His expertise spans IT development and implementation, marketing strategy, legal issues, and budget / financial management.

ADVANCED TARGETED ATTACKS

HAVE PENETRATED **95%** OF ALL NETWORKS*.

THINK YOU'RE IN THE 5%?



You may think your existing security defenses prevent advanced targeted attacks from entering your network and stealing your data. They don't. Advanced attacks easily evade traditional and next generation firewalls, IPS, AV and gateways. Your best defense is **FireEye**. Trusted by the Fortune 500, and over 60 government agencies globally, FireEye is the leader in helping organizations combat advanced malware and targeted APT attacks.

Put a stop to advanced attacks with advanced security. Visit us today at www.FireEye.com/StopAPTs and let us help you close the hole in your network.



*Based on FireEye end-user data
© 2013 FireEye. All rights reserved.



Steel Talon

Mobile Defense System

Steel Talon is a Mobile Defense System intended to do more than scratch the surface of mobile security. The System uses a multitude of techniques to:

- ✔ Detect Intrusion Events
- ✔ Identify Malicious Alterations
- ✔ Transmit Forensic Information

What is Steel Talon?

Steel Talon is uniquely designed to be transparent to the mobile user, and to empower the user's security administrators.

Steel Talon has the ability to remotely disable the phone to protect it from further intrusion as well as automatically and temporarily disable the phone based on intrusion detection.

Locally installed software on the device, and a remote server work in tandem to deliver Steel Talon. An intuitive interface design on both the device and web server allow for immediate adoption.

A KiteWire Product: KiteWire.com

Get more information

Due to the nature of our clients, any further architectural details and system features will require authorization. For more information, please e-mail steeltalon@kitewire.com or call 703-224-8090. This product is not available outside of the United States per the Export Administration Regulations (EAR) of the US Department of Commerce.

How secure is your mobile app?

RIIS offers **code-auditing services**. We'll download your app, decompile it and determine your **security exposure**.



Protect your code.

RIIS understands security, especially mobile security.

Contact us for:

- Code security scanning
- Web Services API Security audits
- Best practice training
- Decompiling workshops on your iOS or Android app

info@riis.com • 248.351.1200 • decompilingandroid.com



Decompiling Android

Written by Godfrey Nolan

Leading expert in exposing risks of Android decompilation; Founder and President of RIIS, wrote the book on decompilation



Research Into Internet Systems

Protection for any device anywhere.

Webroot® SecureAnywhere™ Business solutions deliver the ultimate in endpoint security and protection for all your PCs, smartphones, tablets, servers and virtual machines.

Simplicity - one license covers up to *4 devices per user*.

Lower Costs - as users look to BYOD you won't incur additional costs.

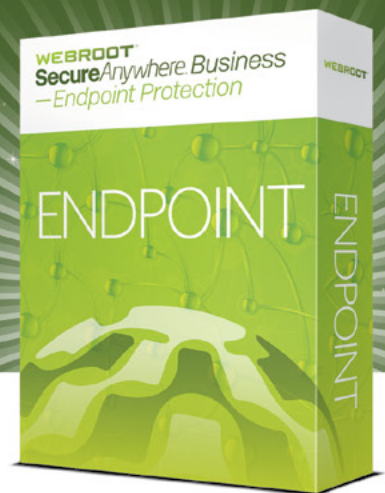
Total Flexibility - a single license covers desktops, laptops, smartphones, tablets, servers and virtual environments.

Powered by the Cloud - secures all your users' devices as well as the infrastructure required to support your business.

Multiple Devices - all managed with a single, intuitive web-based management console that delivers critical visibility to all user devices and every endpoint.

Get Your FREE 30-Day Trial Now!

Visit webroot.com or call 1-800-870-8102



WEBROOT®
SecureAnywhere™
Business