

HACKING RECONNAISSANCE

Finding Vulnerabilities in Your Target Using Nmap

Remember [that scene](#) in *The Matrix* when Trinity uses a realistic [Nmap](#) port scan, followed by an actual [SSH exploit](#) (long since patched) to break into a power company? Well, believe it or not, but that scene is *not* far fetched at all.

If you want to exploit vulnerabilities and root boxes, you'll need to learn how to perform the necessary reconnaissance first. In fact, you will spend far more time researching your target than you will exploiting it. In this article, I am going to show you the first step in doing just that... a security scanner called Nmap.

Port Mapping

Any service running on a server, from HTTP to SSH, runs on ports. Think of a port as a door into and out of the computer, that only answers requests relevant to it. An example would be a web server running on port 80 (HTTP), which would have no idea how to handle an FTP connection request sent to it.

Nmap (**N**etwork **M**apper) scans over those ports telling you everything from what software is running to what version it is. There is even an option to determine the operating system.

Before we get started, I do want to point out something critical. Port mapping, while not illegal on its own, will show up all over the place in targets' server logs. Using a (non-free) VPN or a anonymous network like I2P can help keep you safe and hidden.

Getting Started with Nmap

If you are running [Backtrack](#), you already have Nmap installed, along with its GUI version, [Zenmap](#). Zenmap is nice, but we will be focusing on the command line options for Nmap in this article. On Debian/Ubuntu, simply use:

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

```
user@N5:~$ sudo apt-get install nmap
[sudo] password for user:
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 2 not upgraded.
Need to get 1,623 kB of archives.
After this operation, 7,234 kB of additional disk space will be used.
Get:1 http://ca.archive.ubuntu.com/ubuntu/ oneiric/main nmap i386 5.21-1.1 [1,623 kB]
Fetched 1,623 kB in 5s (282 kB/s)
Selecting previously deselected package nmap.
(Reading database ... 131865 files and directories currently installed.)
Unpacking nmap (from .../nmap_5.21-1.1_i386.deb) ...
Processing triggers for man-db ...
Setting up nmap (5.21-1.1) ...
user@N5:~$
```

Any other distributions that do not already include Nmap may download it [here](#). To get a feel for the software, let's run it with zero options, to see what we can do.

\$ nmap

As you can see, there's a lot of options:

```
Nmap 5.21 ( 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:
  -sL: List Scan - simply list targets to scan
  -sP: Ping Scan - go no further than determining if host is online
  -PN: Treat all hosts as online -- skip host discovery
  -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
  -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/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports consecutively - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
  -sV: Probe open ports to determine service/version info
  --version-intensity <level>: Set from 0 (light) to 9 (try all probes)
  --version-light: Limit to most likely probes (intensity 2)
  --version-all: Try every single probe (intensity 9)
  --version-trace: Show detailed version scan activity (for debugging)
SCRIPT SCAN:
  -sC: equivalent to --script=default
```

```

-O: Enable OS detection
--osscan-limit: Limit OS detection to promising targets
--osscan-guess: Guess OS more aggressively
TIMING AND PERFORMANCE:
Options which take <time> are in milliseconds, unless you append 's'
(seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
-T<0-5>: Set timing template (higher is faster)
--min-hostgroup/max-hostgroup <size>: Parallel host scan group sizes
--min-parallelism/max-parallelism <time>: Probe parallelization
--min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>: Specifies
probe round trip time.
--max-retries <tries>: Caps number of port scan probe retransmissions.
--host-timeout <time>: Give up on target after this long
--scan-delay/--max-scan-delay <time>: Adjust delay between probes
--min-rate <number>: Send packets no slower than <number> per second
--max-rate <number>: Send packets no faster than <number> per second
FIREWALL/IDS EVASION AND SPOOFING:
-f; --mtu <val>: fragment packets (optionally w/given MTU)
-D <decoy1,decoy2[,ME],...>: Cloak a scan with decoys
-S <IP_Address>: Spoof source address
-e <iface>: Use specified interface
-g/--source-port <portnum>: Use given port number
--data-length <num>: Append random data to sent packets
--ip-options <options>: Send packets with specified ip options
--ttl <val>: Set IP time-to-live field
--spoof-mac <mac address/prefix/vendor name>: Spoof your MAC address
--badsum: Send packets with a bogus TCP/UDP/SCTP checksum
--adler32: Use deprecated Adler32 instead of CRC32C for SCTP checksums
OUTPUT:
-oN/-oX/-oS/-oG <file>: Output scan in normal, XML, s|<rIpt kIdDi3,
and Grepable format, respectively, to the given filename.
-oA <basename>: Output in the three major formats at once
-v: Increase verbosity level (use twice or more for greater effect)
-d[level]: Set or increase debugging level (Up to 9 is meaningful)
--reason: Display the reason a port is in a particular state
--open: Only show open (or possibly open) ports
--packet-trace: Show all packets sent and received
--iflist: Print host interfaces and routes (for debugging)
--log-errors: Log errors/warnings to the normal-format output file
--append-output: Append to rather than clobber specified output files
--resume <filename>: Resume an aborted scan
--stylesheet <path/URL>: XSL stylesheet to transform XML output to HTML
--webxml: Reference stylesheet from Nmap.Org for more portable XML
--no-stylesheet: Prevent associating of XSL stylesheet w/XML output

```

While you could write entire books on the full functionality of Nmap (and they have), much of this is beyond the scope of this article. Instead, I will go over some of the more commonly used options. Hopefully this will serve to get your foot in the door with port scanning.

Options, Flags and Settings: Oh My

There is no doubting the sheer size of options here. Let's break it down with what scan techniques are the most useful for us right away.

-sU: [UDP](#) scan. It can be combined with a [TCP](#) scan type such as SYN scan (-sS) to check both protocols during the same run. UDP tends to be slower than TCP scans, but some services are only listening for UDP requests.

-sS: This technique is often referred to as half-open scanning, because you don't open a full TCP connection. You send a SYN packet. A SYN/ACK indicates the port is listening (open), while a RST (reset) means it is not listening on that port.

-O: This technique crafts raw packets attempting to determine the operating system.

-A: This technique tells Nmap to probe for software versions on the target ports AND operating systems.

Nmap in Action

Here we will run a series of port scans on a target web server, making note of the versions and operating systems. Remember, reconnaissance and patience is key to hacking. Let's take a look at Nmap in action as we port scan a web server configured just for this article.

\$ nmap -sS -O 50.22.84.102

```
user@N5:~$ nmap -sS -O 50.22.84.102
You requested a scan type which requires root privileges.
QUITTING!
```

Oops! What happened? The -O switch tells Nmap you wish to perform an operating system fingerprint on the target. In order to do that, Nmap needs to be ran with root privileges in order to craft the raw packets needed for the task. In fact, many scan types require it.

\$ sudo nmap -sS -O 50.22.84.10

```

user@NS:~$ sudo nmap -sS -O 50.22.84.102
[sudo] password for user:
Starting Nmap 5.21 ( http://nmap.org ) at 2017-02-23 10:29 CST
Nmap scan report for sim.simnations.com (50.22.84.102)
Host is up (0.037s latency).
Not shown: 847 closed ports, 141 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
53/tcp    open  domain
80/tcp    open  http
110/tcp   open  pop3
143/tcp   open  imap
443/tcp   open  https
465/tcp   open  smtps
993/tcp   open  imaps
995/tcp   open  pop3s
1306/tcp  open  mysql
8443/tcp  open  https-alt
Device type: general purpose|firewall|proxy server
Running (JUST GUESSING) : Linux 2.6.X (97%), FreeBSD 6.X (91%), Cisco Linux 2.6.X (89%), Riverbed embedded (89%)
Aggressive OS guesses: Linux 2.6.9 - 2.6.27 (97%), Linux 2.6.18 (CentOS 5, x86_64, SMP) (91%), Linux 2.6.9 (91%), FreeBSD 6.2-RELEASE (91%), Linux 2.6.15 - 2.6.26 (91%), Linux 2.6.18 (91%), Linux 2.6.18 (CentOS 5.1, x86) (91%), Linux 2.6.22.1-32.fc6 (x86_64, SMP) (91%), Linux 2.6.28 (90%), Linux 2.6.30 (90%)
No exact OS matches for host (test conditions non-ideal).

OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 35.69 seconds
user@NS:~$

```

Here we can easily see this looks like a normal web server so far. Notice how Nmap attempts to guess the operating system? That's useful when you are looking for an attack vector to exploit.

Open ports - This server is actively accepting TCP connections, UDP datagrams or [SCTP](#) associations on this port. Finding these is often the primary goal of port scanning. 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. Open ports are also interesting for non-security scans because they show services available for use on the network.

Closed ports - A closed port is accessible, 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. Because closed ports are reachable, it may be worth scanning later in case some open up.

Let's try another scan, but this time we want to find out what software is running behind those open ports.

\$ sudo nmap -sS -A 50.22.84.102

```

Starting Nmap 5.21 ( http://nmap.org ) at 2012-02-23 11:01 CST
Nmap scan report for sim.simnations.com (50.22.84.102)
Host is up (0.039s latency).
Not shown: 847 closed ports, 141 filtered ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      PureFTPd
22/tcp    open  ssh      OpenSSH 4.3 (protocol 2.0)
| ssh-hostkey: 1024 76:76:79:ee:4d:b8:24:89:98:29:b4:37:eb:1b:d2:bd (DSA)
| 2048 3e:67:ee:3b:73:97:64:bb:36:5c:c6:2b:91:61:cf:07 (RSA)
33/tcp    open  domain
80/tcp    open  http     Apache/2.2.19 ((Unix) mod_ssl/2.2.19 OpenSSL/0.9.8e-fips-rhel5 DAV/2 mod_auth_passthrough/2.1 mod_bwlimited/1.4 FrontPage/5.0.2635)
|_ html-title: Site doesn't have a title (text/html).
110/tcp   open  pop3     Dovecot pop3d
|_ pop3-capabilities: USER CAPA UIDL TOP OK(K) RESP-CODES PIPELINING STLS SASL(PLAIN LOGIN)
143/tcp   open  imap     Dovecot imapd
|_ imap-capabilities: LOGIN-REFERRALS SORT=DISPLAY AUTH=LOGIN UNSELECT AUTH=PLAIN STARTTLS IMAP4rev1 QUOTA CONDSTORE LIST-STATUS ID SEARCHRES WITHIN CHILDREN LIST-EXTENDED ESORT ESEARCH QRESYNC CONTEXT=SEARCH THREAD=REFS THREAD=REFERENCES I18NLEVEL=1 UIDPLUS NAMESPACE ENABLE SORT LITERAL+ IDLE SASL-IR MULTIAPPEND
443/tcp   open  https    Apache/2.2.19 ((Unix) mod_ssl/2.2.19 OpenSSL/0.9.8e-fips-rhel5 DAV/2 mod_auth_passthrough/2.1 mod_bwlimited/1.4 FrontPage/5.0.2635)
|_ html-title: Site doesn't have a title (text/html).
465/tcp   open  smtp     Exim smtpd 4.69
|_ sslv2: server still supports SSLv2
|_ smtp-commands: EHLO sim.simnations.com Hello 99-118-236-142.lightspeed.cicril.sbcglobal.net [99.118.236.142], SIZE 52428800, PIPELINING, AUTH PLAIN LOGIN, HELP
|_ HELP Commands supported: AUTH HELO EHLO MAIL RCPT DATA NOOP QUIT RSET HELP
993/tcp   open  ssl/imap Dovecot imapd
|_ sslv2: server still supports SSLv2
|_ imap-capabilities: LOGIN-REFERRALS SORT=DISPLAY UNSELECT AUTH=LOGIN AUTH=PLAIN IMAP4rev1 QUOTA CONDSTORE LIST-STATUS ID SEARCHRES WITHIN CHILDREN LIST-EXTENDED ESORT ESEARCH QRESYNC CONTEXT=SEARCH THREAD=REFS THREAD=REFERENCES I18NLEVEL=1 UIDPLUS NAMESPACE ENABLE SORT LITERAL+ IDLE SASL-IR MULTIAPPEND
995/tcp   open  ssl/pop3 Dovecot pop3d
|_ sslv2: server still supports SSLv2
|_ pop3-capabilities: OK(K) CAPA RESP-CODES UIDL PIPELINING USER TOP SASL(PLAIN LOGIN)
3306/tcp   open  mysql    MySQL (unauthorized)
8443/tcp   open  ssl/http Apache/2.0.46 ((Red Hat) mod_ssl/2.0.46 OpenSSL/0.9.7a)
|_ sslv2: server still supports SSLv2
|_ html-title: VZPP Plesk - Log in to Plesk
|_ Requested resource was https://sim.simnations.com:8443/vz/cp/panel/plesk/frameset
Device type: general purpose|firewall|proxy server
Running (JUST GUESSING) : Linux 2.6.X (97%), FreeBSD 6.X (90%), Cisco Linux 2.6.X (89%), Riverbed embedded (89%)

```

Here you can see what software is running and what version. For an example, my web server here is running OpenSSH 4.3 on port 22. If I knew of a vulnerability in that version, I would know this server is exploitable.

Final Thoughts

This is by no means an all inclusive listing of everything Nmap has to offer. I tried to pick and choose the highly relevant portions to give you a feel for its capabilities. You can now add another tool to your ever growing arsenal.

HACK LIKE A PRO

Using the Nmap Scripting Engine (NSE) for Reconnaissance

Those of you who have been reading my posts here for awhile know how much I emphasize [good reconnaissance](#). Novice hackers often jump into a hack/exploit without doing proper recon and either fail or get caught. Experienced and expert hackers know that 70-80 percent of a good and successful hack is dependent upon successful and accurate reconnaissance.

I know I have said it before, but bear with me as I say it again for the newcomers. There is NO SILVER BULLET that succeeds under all circumstances. Long before we ever begin the hack, we have spent hours, days, and maybe months doing reconnaissance. If you aren't willing to do that, you will never be successful in this field of endeavor.

Nmap is one of the few tools that every hacker should be conversant in. Although it is not perfect, it is excellent for [active reconnaissance](#). Although I [discourage the use of Windows for hacking](#), Nmap does have a version for Windows with a nice GUI called [Zenmap](#). You can download it [here](#).

Nmap Scripting Engine (NSE)

I have done [a couple of tutorials](#) on using Nmap, but one thing I have not covered is the scripting engine built into it.

The Nmap scripting engine is one of Nmap's most powerful and, at the same time, most flexible features. It allows users to write their own scripts and share these scripts with other users for the purposes of networking, reconnaissance, etc. These scripts can be used for:

- Network discovery
- More sophisticated and accurate OS version detection
- Vulnerability detection
- Backdoor detection
- Vulnerability exploitation

In this tutorial, we will look at the scripts that have been shared and are built into [Kali](#) (we will write scripts in a future tutorial), and will examine how to use them to do thorough recon on our target, to increase the possibility of success, and reduce the possibilities of frustration... or worse.

Step 1

Fire Up Kali & Open a Terminal

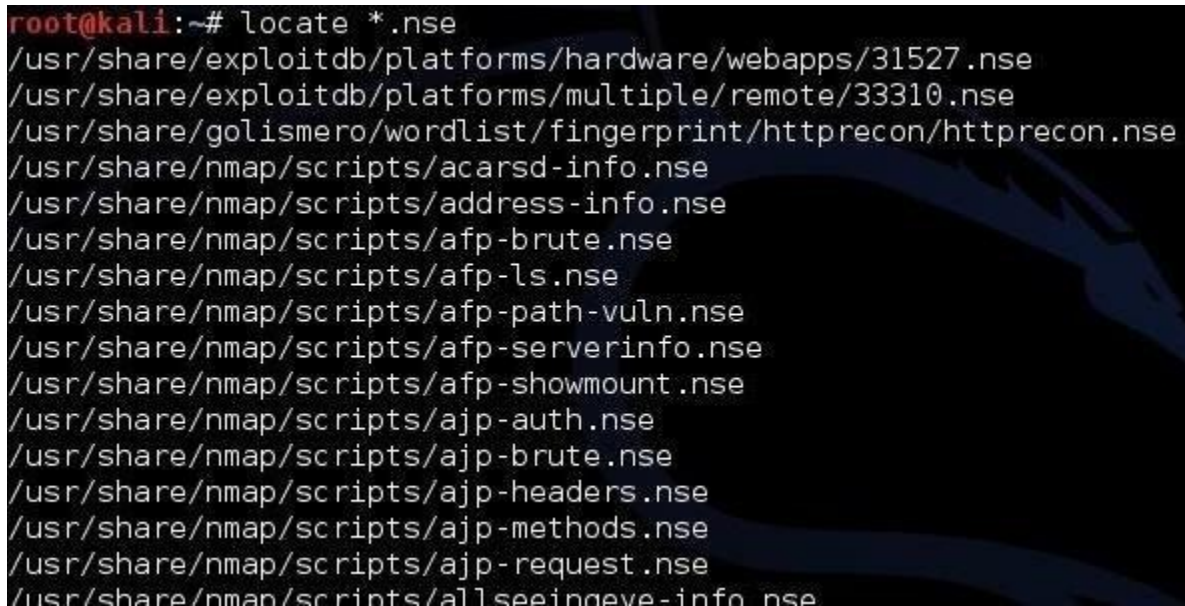
As usual, let's start by firing up Kali and opening a terminal. If you aren't using Kali, but instead one of the many hacking/security distributions such [Bugtraq's Black Window](#), [Security Onion](#), [BackTrack](#), or another, no problem—Nmap is built into ALL of them. You can't really have a security/hacking platform without Nmap.

Step 2

Find the Nmap Scripts

From the terminal, let's look for the Nmap scripts. All of the scripts should end in `.nse` (nmap scripting engine), so we can find the scripts by using the Linux `locate` command with the wildcard `*.nse`. That should find all files ending in `.nse`.

- **kali > locate *.nse**



```
root@kali:~# locate *.nse
/usr/share/exploitdb/platforms/hardware/webapps/31527.nse
/usr/share/exploitdb/platforms/multiple/remote/33310.nse
/usr/share/golismero/wordlist/fingerprint/httprecon/httprecon.nse
/usr/share/nmap/scripts/acarsd-info.nse
/usr/share/nmap/scripts/address-info.nse
/usr/share/nmap/scripts/afp-brute.nse
/usr/share/nmap/scripts/afp-ls.nse
/usr/share/nmap/scripts/afp-path-vuln.nse
/usr/share/nmap/scripts/afp-serverinfo.nse
/usr/share/nmap/scripts/afp-showmount.nse
/usr/share/nmap/scripts/ajp-auth.nse
/usr/share/nmap/scripts/ajp-brute.nse
/usr/share/nmap/scripts/ajp-headers.nse
/usr/share/nmap/scripts/ajp-methods.nse
/usr/share/nmap/scripts/ajp-request.nse
/usr/share/nmap/scripts/allseeingeve-info.nse
```

As you can see in the screenshot above, our terminal displays hundreds of Nmap scripts.

Step 3 Finding Vulnerability Scanning Scripts

Among the most useful to us are the vulnerability scanning scripts. These scripts are usually designed to find a specific vulnerability or type of vulnerability that we can then come back later and exploit. To locate those scripts that we can use for vulnerability scanning, we can type:

- **kali> locate *vuln*.nse**

```
/usr/share/nmap/scripts/atp-path-vuln.nse
/usr/share/nmap/scripts/ftp-vuln-cve2010-4221.nse
/usr/share/nmap/scripts/http-huawei-hg5xx-vuln.nse
/usr/share/nmap/scripts/http-iis-webdav-vuln.nse
/usr/share/nmap/scripts/http-vmware-path-vuln.nse
/usr/share/nmap/scripts/http-vuln-cve2009-3960.nse
/usr/share/nmap/scripts/http-vuln-cve2010-0738.nse
/usr/share/nmap/scripts/http-vuln-cve2010-2861.nse
/usr/share/nmap/scripts/http-vuln-cve2011-3192.nse
/usr/share/nmap/scripts/http-vuln-cve2011-3368.nse
/usr/share/nmap/scripts/http-vuln-cve2012-1823.nse
/usr/share/nmap/scripts/http-vuln-cve2013-0156.nse
/usr/share/nmap/scripts/http-vuln-zimbra-lfi.nse
/usr/share/nmap/scripts/mysql-vuln-cve2012-2122.nse
/usr/share/nmap/scripts/rdp-vuln-ms12-020.nse
/usr/share/nmap/scripts/rmi-vuln-classloader.nse
/usr/share/nmap/scripts/samba-vuln-cve-2012-1182.nse
/usr/share/nmap/scripts/smb-check-vulns.nse
/usr/share/nmap/scripts/smb-vuln-ms10-054.nse
/usr/share/nmap/scripts/smb-vuln-ms10-061.nse
/usr/share/nmap/scripts/smtp-vuln-cve2010-4344.nse
/usr/share/nmap/scripts/smtp-vuln-cve2011-1720.nse
/usr/share/nmap/scripts/smtp-vuln-cve2011-1764.nse
```

As you can see, it returned a few vulnerability scanning scripts. I have highlighted one I want to use next, namely *smb-check-vulns.nse*. This script will check the system to see whether it has any of the well-known SMB vulnerabilities such as [MS08-067](#).

Step 4 Running the Script

The basic syntax for running these scripts is this:

- **nmap --script <scriptname> <host ip>**

Let's try running the SMB vulnerability checking script against an internal LAN host.

- **kali> nmap --script smb-check-vulns-nse 192.168.1.121**

```
1035/tcp open  multidropper
1433/tcp open  ms-sql-s
5800/tcp open  vnc-http
5900/tcp open  vnc
MAC Address: 00:0C:29:18:6B:DB (VMware)

Host script results:
| smb-check-vulns:
|   MS08-067: CHECK DISABLED (add '--script-args=unsafe=1' to run)
|   Conficker: Likely CLEAN
|   regsvc DoS: CHECK DISABLED (add '--script-args=unsafe=1' to run)
|   SMBv2 DoS (CVE-2009-3103): CHECK DISABLED (add '--script-args=unsafe=1' to r
un)
|   MS06-025: CHECK DISABLED (add '--script-args=unsafe=1' to run)
|_  MS07-029: CHECK DISABLED (add '--script-args=unsafe=1' to run)

Nmap done: 1 IP address (1 host up) scanned in 14.11 seconds
root@kali:~#
```

When we do so, we can see that it returns some errors and suggests that we add `--script-args=unsafe=1` to our command. I did so below and, in this case, added `-p445` so that the script focuses upon just the SMB port 445.

- **kali> nmap --script-args=unsafe=1 --script smb-check-vulns.nse -p445**

192.168.1.121

```
root@kali:~# nmap --script-args=unsafe=1 --script smb-check-vulns.nse -p445 192.
168.1.121
```

Now, when I run the command, I get much more useful results.

```
root@kali:~# nmap --script-args=unsafe=1 --script smb-check-vulns.nse -p445 192.168.1.121

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-21 16:42 MST
Nmap scan report for 192.168.1.121
Host is up (0.0027s latency).
PORT      STATE SERVICE
445/tcp    open  microsoft-ds
MAC Address: 00:0C:29:18:6B:DB (VMware)

Host script results:
| smb-check-vulns:
|   MS08-067: VULNERABLE
|   Conficker: Likely CLEAN
|   SMBv2 DoS (CVE-2009-3103): NOT VULNERABLE
|   MS06-025: NOT VULNERABLE
|_  MS07-029: NO SERVICE (the Dns Server RPC service is inactive)

Nmap done: 1 IP address (1 host up) scanned in 18.74 seconds
```

As you can see, it tells me that MS08-067 is vulnerable, so now I know I can use that module in [Metasploit](#) to exploit that system!

Step 5 Getting Help with a Script

With these hundreds of scripts, we may need some help in determining what they do and how they work. For instance, if we scroll down to the "http" section of the scripts, we will see a script named:

- **http-vuln-cve2013-0156.nse**

To determine what this script does, we can retrieve its help file by typing:

- **nmap -script-help http-vuln-cve2013-0156.nse**

When we do so, we will retrieve its help file like that below.


```
root@kali:~# nmap -script-help http-vuln-cve2013-0156.nse

Starting Nmap 6.46 ( http://nmap.org ) at 2014-11-20 15:50 MST

http-vuln-cve2013-0156
Categories: exploit vuln
http://nmap.org/nsedoc/scripts/http-vuln-cve2013-0156.html
  Detects Ruby on Rails servers vulnerable to object injection, remote command e
xecutions and denial of service attacks. (CVE-2013-0156)

  All Ruby on Rails versions before 2.3.15, 3.0.x before 3.0.19, 3.1.x before 3.
1.10, and 3.2.x before 3.2.11 are vulnerable. This script
  sends 3 harmless yaml payloads to detect vulnerable installations. If the malf
ormed object receives a status 500 response, the server
  is processing YAML objects and therefore is likely vulnerable.

References:
* https://community.rapid7.com/community/metasploit/blog/2013/01/10/exploiting
-ruby-on-rails-with-metasploit-cve-2013-0156',
* https://groups.google.com/forum/?fromgroups=#!msg/rubyonrails-security/61bkg
```

Note that this script is designed to find the CVE2013-0156 vulnerability, which is a vulnerability in Ruby on Rails. Ruby on Rails is very popular open-source web design framework that is behind millions of database driven web apps, so this vulnerability is likely to still be out there in thousands of websites. Happy hunting!

The Nmap scripting engine is a powerful item in our arsenal of hacking tools that can be tailored to a multitude of tasks. In future posts, I will explore more of its capabilities and show you how to write your own Nmap scripts. So keep coming back, my tenderfoot hackers!