

# PWN DEFEND

## Hacking 101

BASIC NETWORK ENUMERATION

Version 0.91 - Draft

#pwnDefend

# Welcome

Welcome to the first short course in our hacking 101 series for pwnDefend! The intention behind this is to give people new to the security testing/hacking world some basic information on how to perform network enumeration using common tools such as nmap.

This series has been created by Daniel Card (UK\_Daniel\_Card) and is aimed to educate and help people with some basic tools, techniques and practises.

**Please only use this information for good! Ensure you use these tools where you either own the assets or have the owner permission! Check local laws and don't break them! #hack4good**

# Introduction

This course aims to teach the basics of network enumeration

- By the end of the course you should be familiar with port scanning techniques and standard networking tools such as:
  - NMAP
  - UNICORNSCAN
  - MASSSCAN

# Learning Goals

- Understand port scanning and how it's useful for reconnaissance/enumeration
- Understand the basic syntax of common port scanning tools

# Course Requirements

Whilst you should be able to follow the content here without a huge level of experience it's advisable to have some knowledge in the following areas:

- OS fundamentals LINUX and WINDOWS
- Basic TCP/IP Network knowledge
- Experience using a hypervisor such as ORACLE Virtual Box, Hyper-V, VMware player etc.



# Why we care?

- Enumeration gives takes us from being essentials in the dark through to shining a light behind a surface to see if we can spot any entry points or cracks in the outer layers.
- The most important aspect of any attack is enumeration, it may not be as glamorous as popping shells but without it your just bopping about in the ocean of the network/internet.

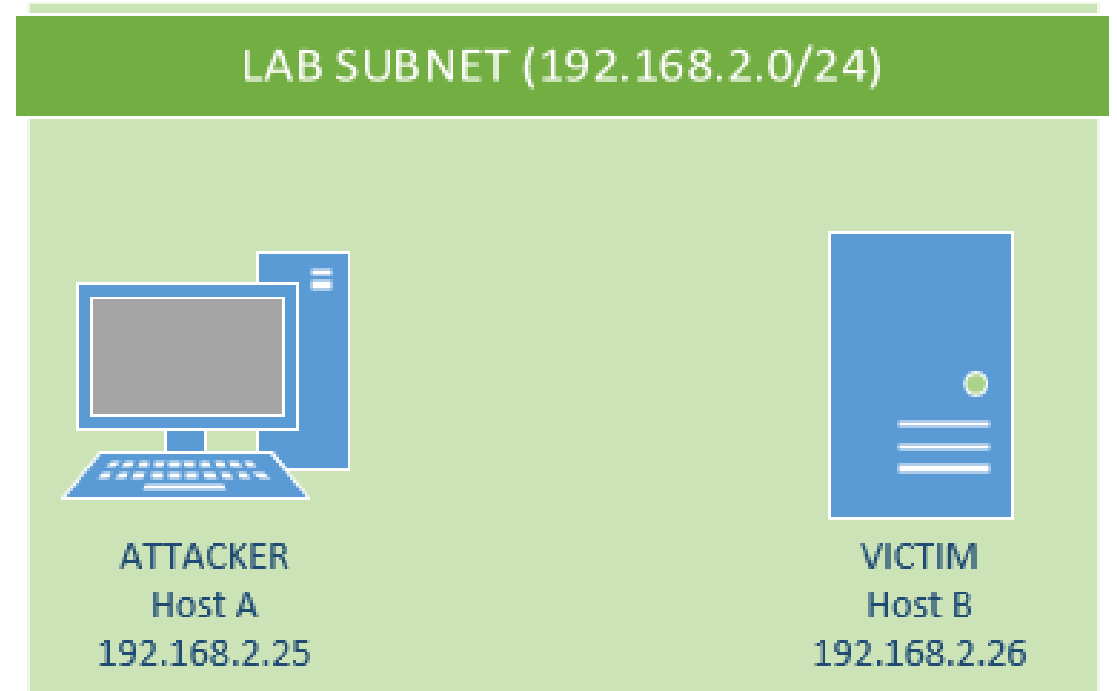
# Situation

There are two likely scenarios you will find yourselves in:

- You are internet facing and your target area is a public IP space
- You are on a network either directly or via a pivot

# Topology

In this course we are using a simple setup with two virtual machines. This can easily be setup with moderate hardware using Oracle Virtual Box, Hyper-V or VMware workstation etc.





# Background Knowledge

It helps to have a basic understanding of TCP/IP. Especially the 3 way handshake of TCP and the 4 four-step FIN. But we mustn't forget there are more protocols than TCP e.g. UDP, ICMP, GRE etc.

<https://study-ccna.com/tcp-three-way-handshake/>

Correct Three Way sequence =

1. (Host A) SYN
2. (Host B) SYN,ACK
3. (Host A) ACK

Once the handshake and data transmission is complete a four-step fin is conducted:

1. (Host A) FIN
2. (Host B) ACK
3. (Host B) FIN
4. (Host A) ACK

Whilst we aren't going to go into detail on these, it's important to know what a TCP Connect, ACK and SYN scan do and how they work in relation to the above.

# Tools

- To complete this module you will need the following:
  - A KALI Instance
  - Access to the internet (for git downloads)
  - A virtual machine to port scan (in the module I've used Windows Server 2019 as the target)

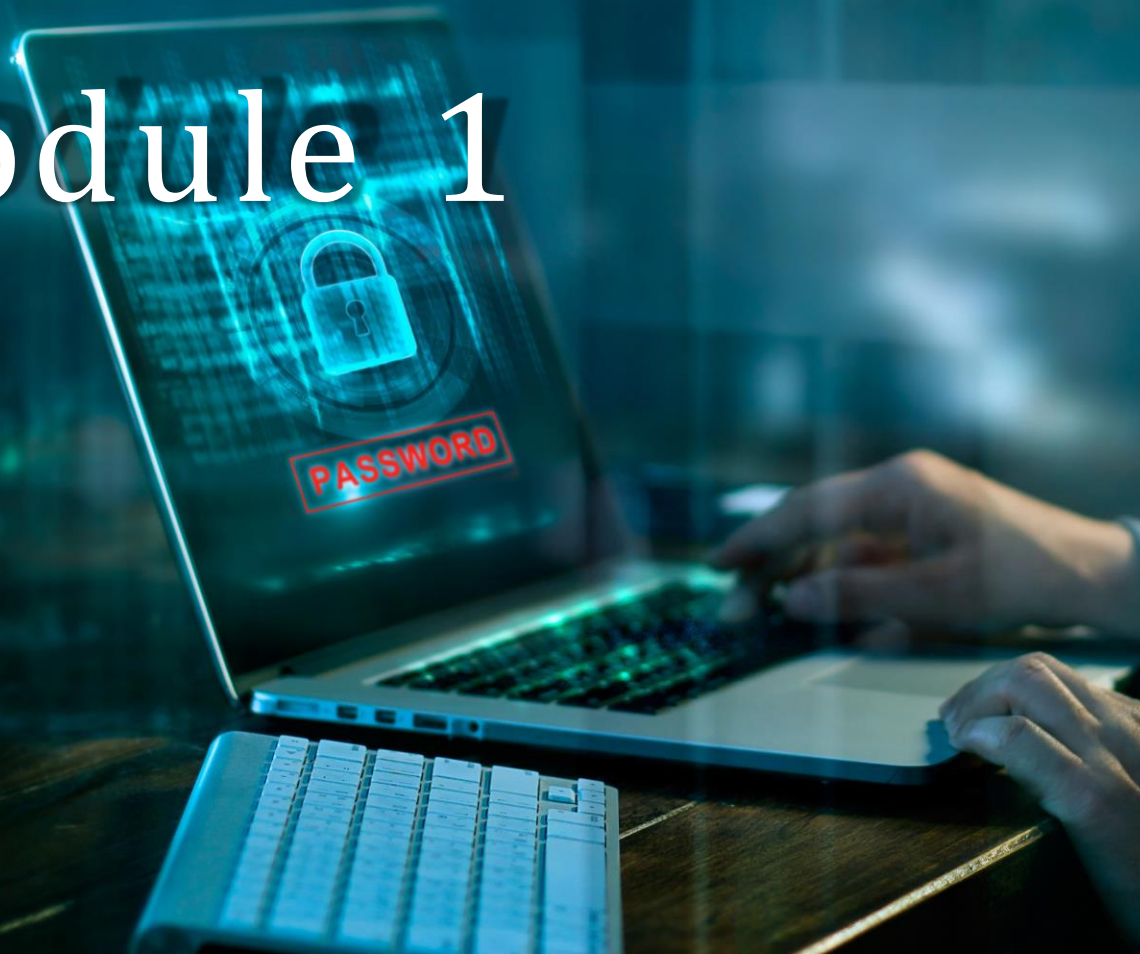
# Tools

- NMAP (<https://nmap.org>)
- Unicornscan (<https://tools.kali.org/information-gathering/unicornscan>)
- MASSCAN (<https://github.com/robertdavidgraham/masscan>)

# PWN DEFEND

## Module 1

NMAP



# Network Scanning with NMAP

Nmap is one of the swiss army knives of the security tester! It has a seriously huge range of functionality.

The basic functionality we are going to look at in this course are as follows:

- Port scanning
- Fingerprinting
- NMAP Scripting Environment

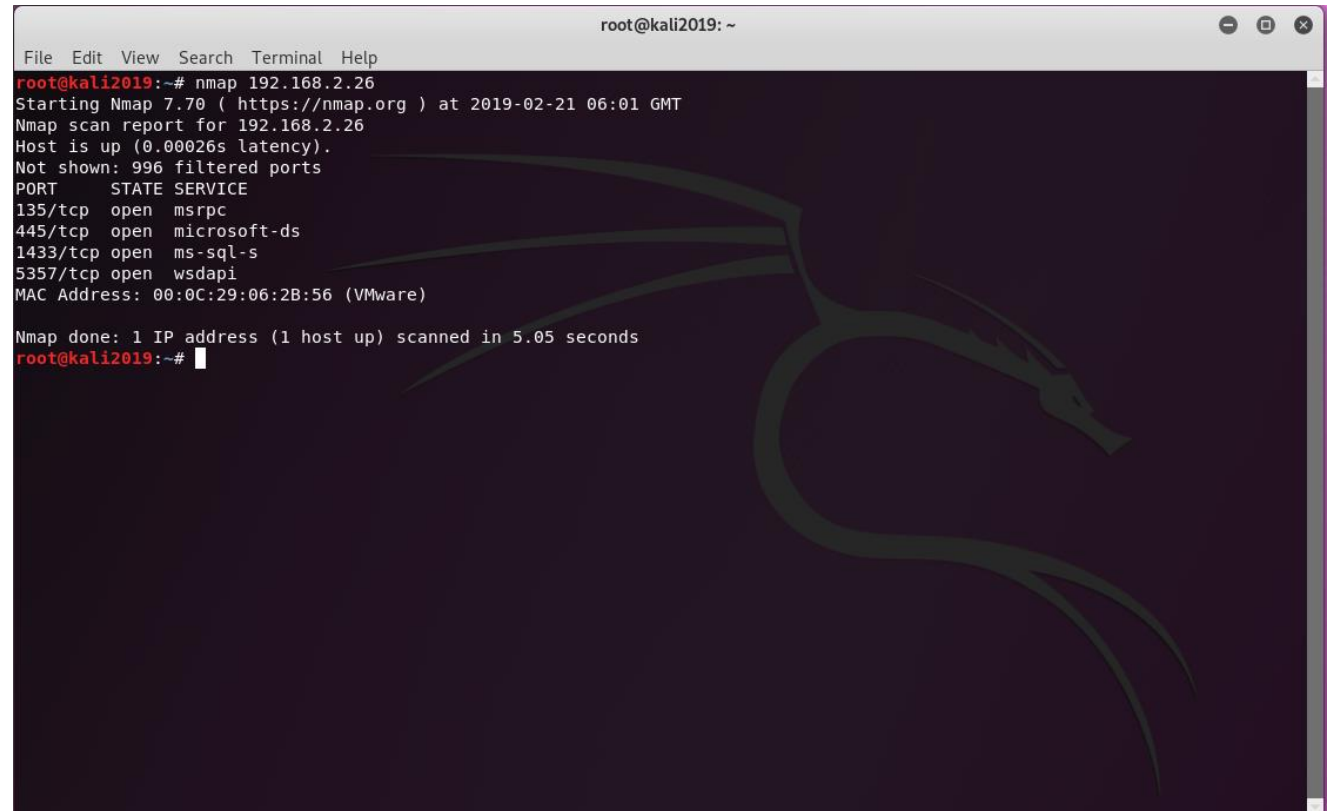
# NMAP Basic Command Usage

At it's most simple

- nmap [Target\_IP]

e.g. nmap 192.168.2.26

This will run a basic scan and give us some key network intel



```
root@kali2019: ~  
File Edit View Search Terminal Help  
root@kali2019:~# nmap 192.168.2.26  
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-21 06:01 GMT  
Nmap scan report for 192.168.2.26  
Host is up (0.00026s latency).  
Not shown: 996 filtered ports  
PORT      STATE SERVICE  
135/tcp    open  msrpc  
445/tcp    open  microsoft-ds  
1433/tcp   open  ms-sql-s  
5357/tcp   open  wsdapi  
MAC Address: 00:0C:29:06:2B:56 (VMware)  
  
Nmap done: 1 IP address (1 host up) scanned in 5.05 seconds  
root@kali2019:~#
```

# The default nmap scan

The default nmap scan does the following:

- Scans TCP 1-1000
- Scans ports in a random order

<https://nmap.org/book/man-port-specification.html>

# Common Scans

- SYN Scan
- TCP Scan
- UDP Scan

When probing a target we need to consider it may be defended. In fact it advisable to assume a target it defended and act appropriately to avoid tripping an intrusion prevention or detection control.



# NMAP scans

SYN Scan with full port range, fast timing and all output formats with OS detection

*`nmap -sS -sV -T4 -p- -O -A -oA module1 192.168.2.26`*

```
root@kali2019:~# nmap -sS -sV -T4 -O -A -oA module1 192.168.2.26
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-21 07:00 GMT
Nmap scan report for 192.168.2.26
Host is up (0.00075s latency).
Not shown: 996 filtered ports
PORT      STATE SERVICE          VERSION
135/tcp    open  msrpc            Microsoft Windows RPC
445/tcp    open  microsoft-ds?
1433/tcp   open  ms-sql-s         Microsoft SQL Server 15.00.1200.00
| ms-sql-ntlm-info:
|   Target Name: WIN-HP2VHVL3L9P
|   NetBIOS_Domain_Name: WIN-HP2VHVL3L9P
|   NetBIOS_Computer_Name: WIN-HP2VHVL3L9P
|   DNS_Domain_Name: WIN-HP2VHVL3L9P
|   DNS_Computer_Name: WIN-HP2VHVL3L9P
|   Product_Version: 10.0.17763
|_  ssl-cert: Subject: commonName=SSL_Self_Signed_Fallback
|   Not valid before: 2019-02-20T05:39:13
|   Not valid after: 2049-02-20T05:39:13
|_  ssl-date: 2019-02-21T07:00:26+00:00; 0s from scanner time.
5357/tcp   open  http             Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_  http-server-header: Microsoft-HTTPAPI/2.0
|_  http-title: Service Unavailable
MAC Address: 00:0C:29:06:2B:56 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
OS fingerprint not ideal because: Missing a closed TCP port so results incomplete
No OS matches for host
Network Distance: 1 hop
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_  ms-sql-info:
|     192.168.2.26:1433:
|       Version:
|       name: Microsoft SQL Server
```

# NMAP UDP Scan

Now UDP scans are slow and due to the nature of the UDP protocol it's possible to get some false positives and UDP scans take some time!

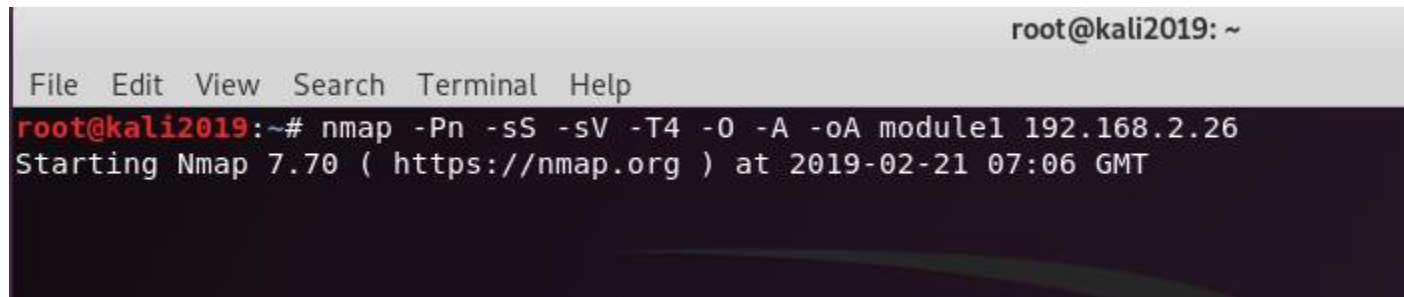
```
nmap -Pn -sU -sV -T4 -O -A -oA module1-UDP 192.168.2.26
```

*Expect a top 1000 UDP scan to take ~1 hour on a well connected (Ethernet) system*

# Disable Ping

Often a target won't respond to ICMP ECHO (ping) in which case you need to prepend the following to your command line:

*-Pn*

A terminal window screenshot from a Kali Linux machine. The title bar shows 'root@kali2019: ~'. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The command prompt shows 'root@kali2019:~# nmap -Pn -sS -sV -T4 -O -A -oA module1 192.168.2.26'. The output shows 'Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-21 07:06 GMT'.

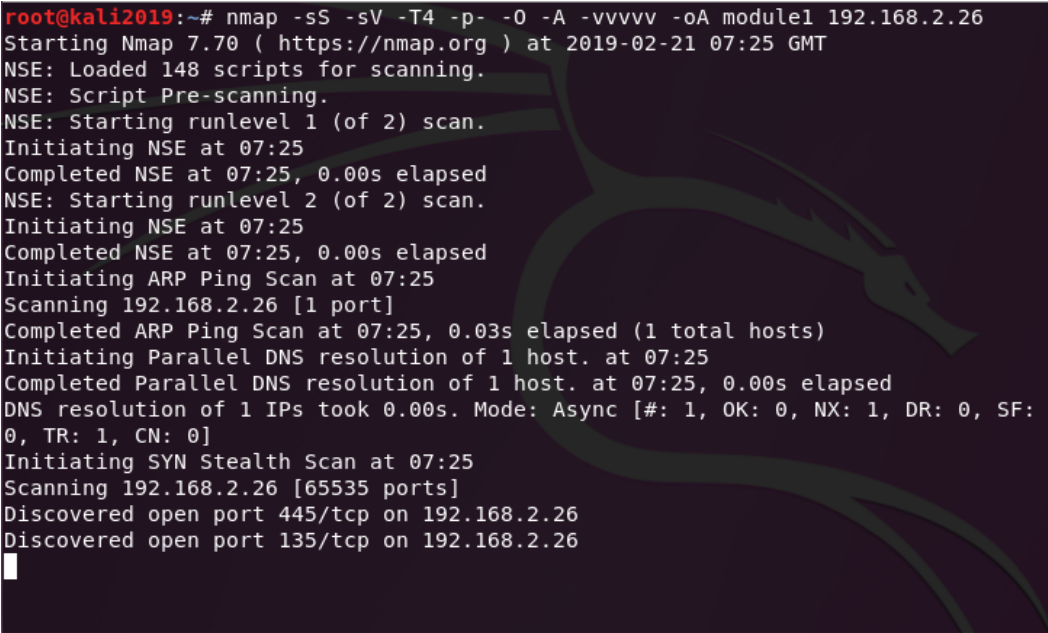
```
root@kali2019: ~
File Edit View Search Terminal Help
root@kali2019:~# nmap -Pn -sS -sV -T4 -O -A -oA module1 192.168.2.26
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-21 07:06 GMT
```

# Verbosity

Up until now you may have noticed, especially on a UDP scan that watching the scan complete is a little bland, you may even think it has crashed sometimes! So to change this we can increase the verbosity, not only does that improve our experience it can also help with troubleshooting. To do this add the following to the command line:

`-v`

You can increase the number of V's to change the level of verbosity



```
root@kali2019:~# nmap -sS -sV -T4 -p- -O -A -vvvvv -oA module1 192.168.2.26
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-21 07:25 GMT
NSE: Loaded 148 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 2) scan.
Initiating NSE at 07:25
Completed NSE at 07:25, 0.00s elapsed
NSE: Starting runlevel 2 (of 2) scan.
Initiating NSE at 07:25
Completed NSE at 07:25, 0.00s elapsed
Initiating ARP Ping Scan at 07:25
Scanning 192.168.2.26 [1 port]
Completed ARP Ping Scan at 07:25, 0.03s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 07:25
Completed Parallel DNS resolution of 1 host. at 07:25, 0.00s elapsed
DNS resolution of 1 IPs took 0.00s. Mode: Async [#: 1, OK: 0, NX: 1, DR: 0, SF:
0, TR: 1, CN: 0]
Initiating SYN Stealth Scan at 07:25
Scanning 192.168.2.26 [65535 ports]
Discovered open port 445/tcp on 192.168.2.26
Discovered open port 135/tcp on 192.168.2.26
```

# NMAP Scripting Engine (NSE)

NMAP contains an extensible script engine, it has hundreds of scripts out of the box! It can do things like scan for missing HTTP headers, analyse SSL/TLS strength

It's a huge subject so get your google/browsing on and get reading, better yet practise in the lab! The following command line will scan using all scripts that include vuln in them!

```
nmap -sV --script=*vuln* 192.168.2.26
```

<https://nmap.org/book/nse.html>

# Evading IPS/IDS

Often targets will have some form of network protection, nmap has a host of options for evasion which include:

- Proxies
- ACK Scans
- Decoys
- MAC spoofing
- Fragmentation
- Randomisation
- Timing



# Evasion Example

The following example  
creates packet  
fragments to evade  
signature detection:

```
nmap -sT -f --top-ports  
20 192.168.2.26
```

We can increase  
fragmentation using  
more *-ff*

```
root@kali2019:~# nmap -sT -f --top-ports 20 192.168.2.26
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-21 08:57 GMT
Stats: 0:00:00 elapsed; 0 hosts completed (0 up), 1 undergoing ARP Ping Scan
Parallel DNS resolution of 1 host. Timing: About 0.00% done
Nmap scan report for 192.168.2.26
Host is up (0.00060s latency).

PORT      STATE      SERVICE
21/tcp    filtered  ftp
22/tcp    filtered  ssh
23/tcp    filtered  telnet
25/tcp    filtered  smtp
53/tcp    filtered  domain
80/tcp    filtered  http
110/tcp   filtered  pop3
111/tcp   filtered  rpcbind
135/tcp   open       msrpc
139/tcp   filtered  netbios-ssn
143/tcp   filtered  imap
443/tcp   filtered  https
445/tcp   open       microsoft-ds
993/tcp   filtered  imaps
995/tcp   filtered  pop3s
1723/tcp  filtered  pptp
3306/tcp  filtered  mysql
3389/tcp  filtered  ms-wbt-server
5900/tcp  filtered  vnc
8080/tcp  filtered  http-proxy
MAC Address: 00:0C:29:06:2B:56 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.55 seconds
```



# Pro Tip

Whilst this will be covered in a later course there's an awesome trick you can do with NMAP which uses the -sV mode and searchsploit combined!

```
nmap -p- -sV -oX new.xml  
192.168.2.26; searchsploit --nmap  
new.xml
```

This will highlight if any enumerated services may have vulnerabilities in exploit DB!

```
root@kali2019:~# nmap -p- -sV -oX new.xml 192.168.2.26; searchsploit --nmap new.xml
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-21 07:36 GMT
Nmap scan report for 192.168.2.26
Host is up (0.00045s latency).
Not shown: 65528 filtered ports
PORT      STATE SERVICE        VERSION
135/tcp    open  msrpc          Microsoft Windows RPC
445/tcp    open  microsoft-ds?
1433/tcp   open  ms-sql-s       Microsoft SQL Server
5357/tcp   open  http           Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
5985/tcp   open  http           Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49666/tcp  open  msrpc          Microsoft Windows RPC
49668/tcp  open  msrpc          Microsoft Windows RPC
1 service unrecognized despite returning data. If you know the service/version, please submit the
following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port1433-TCP:V=7.70%I=7%D=2/21%Time=5C6E5577%P=x86_64-pc-linux-gnu%r(ms
SF:-sql-s,25,"\\x04\\x01\\0%\\0\\x01\\0\\0\\x15\\x06\\x01\\0\\x1b\\x01\\x02\\0\\x1
SF:c\\0\\x01\\x03\\0\\x1d\\0\\0\\x0f\\x0f\\x04\\xb0\\0\\0\\0");
MAC Address: 00:0C:29:06:2B:56 (VMware)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 159.00 seconds
[i] SearchSploit's XML mode (without verbose enabled). To enable: searchsploit -v --xml...
[i] Reading: 'new.xml'

[i] /usr/bin/searchsploit -t microsoft windows rpc
-----
Exploit Title | Path
(-----)
Microsoft Windows - 'Lsasrv.dll' RPC R | exploits/windows/remote/293.c
Microsoft Windows - 'RPC DCOM' Long Fi | exploits/windows/remote/100.c
Microsoft Windows - 'RPC DCOM' Remote | exploits/windows/remote/64.c
Microsoft Windows - 'RPC DCOM' Remote | exploits/windows/remote/69.c
Microsoft Windows - 'RPC DCOM' Remote | exploits/windows/remote/70.c
Microsoft Windows - 'RPC DCOM' Remote | exploits/windows/remote/76.c
Microsoft Windows - 'RPC DCOM' Scanner | exploits/windows/remote/97.c
Microsoft Windows - 'RPC DCOM2' Remote | exploits/windows/remote/103.c
Microsoft Windows - 'RPC2' Universal / | exploits/windows/remote/109.c
Microsoft Windows - DCE-RPC svcctl Cha | exploits/windows/dos/3453.py
Microsoft Windows - DCOM RPC Interface | exploits/windows/remote/22917.txt
```



# OneTwoPunch

OneTwoPunch combines unicornscan and nmap to rapidly scan and enumerate all 65K ports

```
root@kali2019:/pentest/onetwopunch# ./onetwopunch.sh
```

onetwopunch  
by superkojiman

Usage: ./onetwopunch.sh -t targets.txt [-p tcp/udp/all] [-i interface] [-n nmap-options] [-h]

- h: Help
- t: File containing ip addresses to scan. This option is required.
- p: Protocol. Defaults to tcp
- i: Network interface. Defaults to eth0
- n: NMAP options (-A, -O, etc). Defaults to no options.

```
root@kali2019:/pentest/onetwopunch#
```

# Usage

Running `./onetwopunch.sh` will show the usage options. We need to create a target file:

```
echo 192.168.2.26 > target.txt
```

then run:

```
./onetwopunch.sh -t target.txt -  
p all
```

```

root@kali2019:/pentest/onetwopunch# ./onetwopunch.sh

  _____
 /  _  _  _  \
(  _  _  _  ) C(ò_ó")
 \  _  _  _  /
  \_____/

  _____
 /  _  _  _  \
(  _  _  _  )
 \  _  _  _  /
  \_____/

  _____
 /  _  _  _  \
(  _  _  _  )
 \  _  _  _  /
  \_____/

by superkojiman

Usage: ./onetwopunch.sh -t targets.txt [-p tcp/udp/all] [-i interface] [-n nmap-options] [-h]
-h: Help
-t: File containing ip addresses to scan. This option is required.
-p: Protocol. Defaults to tcp
-i: Network interface. Defaults to eth0
-n: NMAP options (-A, -O, etc). Defaults to no options.
root@kali2019:/pentest/onetwopunch# echo 192.168.2.26 > target.txt
root@kali2019:/pentest/onetwopunch# ./onetwopunch.sh -t target.txt -p all

  _____
 /  _  _  _  \
(  _  _  _  ) C(ò_ó")
 \  _  _  _  /
  \_____/

  _____
 /  _  _  _  \
(  _  _  _  )
 \  _  _  _  /
  \_____/

  _____
 /  _  _  _  \
(  _  _  _  )
 \  _  _  _  /
  \_____/

by superkojiman

[+] Protocol : all
[+] Interface: eth0
[+] Nmap opts: -sV
[+] Targets  : target.txt
[+] Scanning 192.168.2.26 for all ports...
[+] Obtaining all open TCP ports using unicornscan...
[+] unicornscan -i eth0 -mT 192.168.2.26:a -l /root/.onetwopunch/udir/192.168.2.26-tcp.txt

```



# OneTwoPunch

Now we have seen NMAP and UNICORNSCAN, what if you were to combine them? well here we have OneTwoPunch!

If you want to run a scan of both TCP and UDP at the same time across all 65K ports try this:

<https://github.com/superkojiman/onetwopunch>

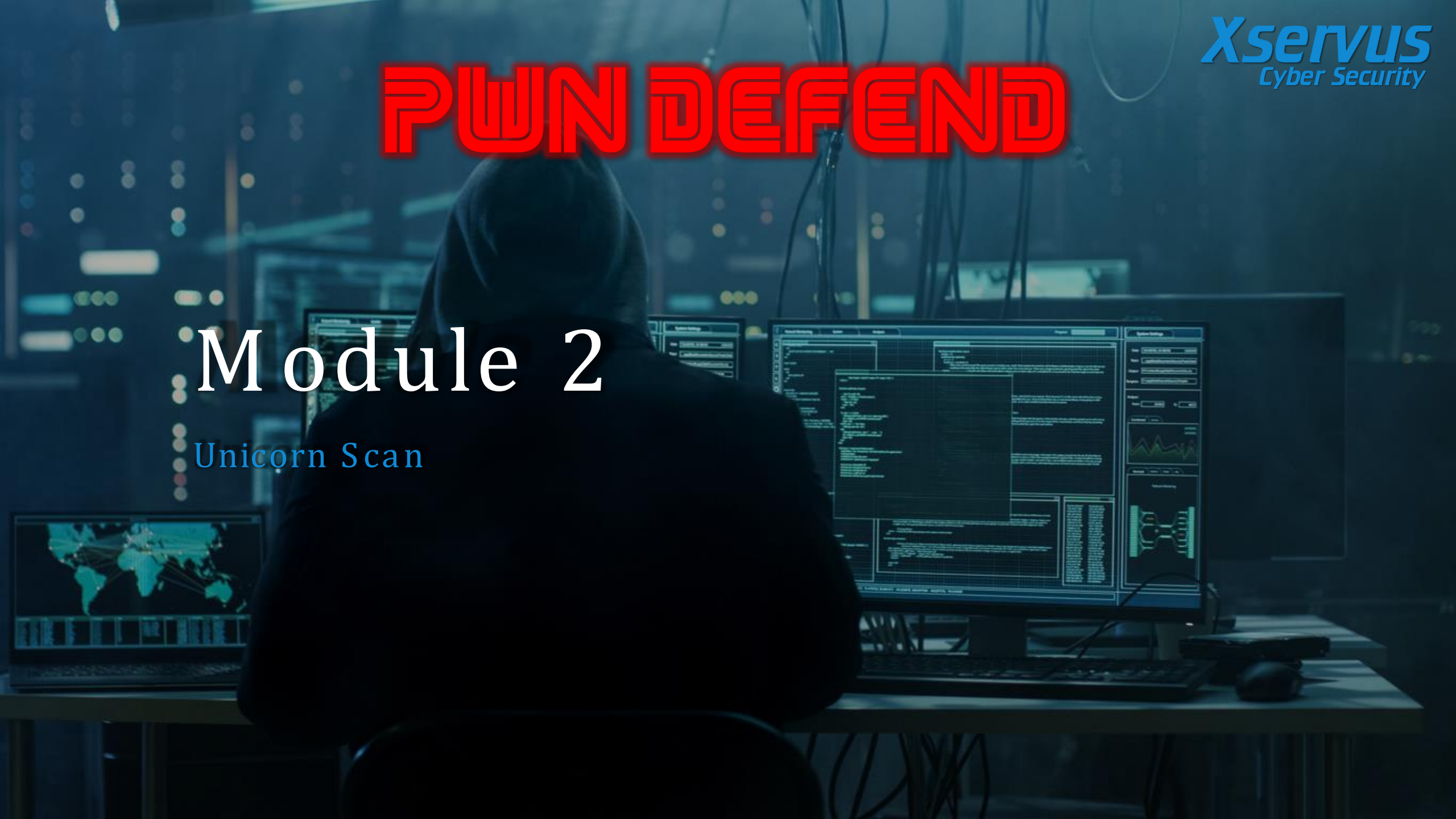
Thanks to Harold Rodriguez (superkojiman) for creating!



# PWN DEFEND

## Module 2

Unicorn Scan



# Unicorn Scan

Unicorn scan has a range of features and can be faster to get results than nmap. It has its own TCP/IP stack so if your scanning from a windows host your going to see a big speed improvement. Because it has its own TCP/IP stack it can do some clever OS spoofing as well!

```
root@kali2019:~# unicornscan 192.168.2.26
TCP open          epmap[ 135]          from 192.168.2.26  ttl 128
TCP open          microsoft-ds[ 445]      from 192.168.2.26  ttl 128
TCP open          ms-sql-s[ 1433]       from 192.168.2.26  ttl 128
root@kali2019:~#
```

<https://tools.kali.org/information-gathering/unicornscan>



# Command Line Parameters

Unicorn (like nmap) has a huge range of command line options.

To view these run:

*unicornscan --help*

Simple SYN Scan

*unicornscan 192.168.2.26*

UDP Scan

*unicornscan -r300 -mU  
192.168.2.26*

```
root@kali2019:~# unicornscan --help
unicornscan (version 0.4.7)
usage: unicornscan [options] 'b:B:cd:De:EFg:hHi:Ij:l:L:m:M:o:p:P:q:Or:R:s:St:T:u:Uw:W:vVzZ:' ] X.X.X.X/YY:S-E
-b, --broken-crc      *set broken crc sums on [T]ransport layer, [N]etwork layer, or both[TN]
-B, --source-port     *set source port? or whatever the scan module expects as a number
-c, --proc-duplicates process duplicate replies
-d, --delay-type       *set delay type (numeric value, valid options are '1:tsc 2:gtod 3:sleep')
-D, --no-defpayload    no default Payload, only probe known protocols
-e, --enable-module    *enable modules listed as arguments (output and report currently)
-E, --proc-errors      for processing 'non-open' responses (icmp errors, tcp rst...)
-F, --try-frags
-G, --payload-group    *payload group (numeric) for tcp/udp type payload selection (default all)
-h, --help            help
-H, --do-dns           resolve hostnames during the reporting phase
-i, --interface        *interface name, like eth0 or fxp1, not normally required
-I, --immediate        immediate mode, display things as we find them
-j, --ignore-seq       *ignore 'A'll, 'R'eset sequence numbers for tcp header validation
-l, --logfile          *write to this file not my terminal
-L, --packet-timeout   *wait this long for packets to come back (default 7 secs)
-m, --mode             *scan mode, tcp (syn) scan is default, U for udp T for tcp 'sf' for tcp connect scan and A for arp

for -mT you can also specify tcp flags following the T like -mTsFpU for example
that would send tcp syn packets with (NO Syn|FIN|NO Push|URG)

-M, --module-dir       *directory modules are found at (defaults to /usr/lib/unicornscan/modules)
-o, --format           *format of what to display for replies, see man page for format specification
-p, --ports            global ports to scan, if not specified in target options
-P, --pcap-filter       *extra pcap filter string for reciever
-q, --covertiness      *covertiness value from 0 to 255
-Q, --quiet            dont use output to screen, its going somewhere else (a database say...)
-r, --pps              *packets per second (total, not per host, and as you go higher it gets less accurate)
-R, --repeats          *repeat packet scan N times
-s, --source-addr      *source address for packets 'r' for random
-S, --no-shuffle        do not shuffle ports
-t, --ip-ttl           *set TTL on sent packets as in 62 or 6-16 or r64-128
-T, --ip-tos           *set TOS on sent packets
-u, --debug            *debug mask
-U, --no-openclosed    dont say open or closed
-w, --safe             *write pcap file of recieved packets
-W, --fingerprint      *OS fingerprint 0=cisco(def) 1=openbsd 2=WindowsXP 3=p0fsendsyn 4=FreeBSD 5=nmap
                        6=linux 7:strangetcp
-v, --verbose          verbose (each time more verbose so -vvvvv is really verbose)
-V, --version          display version
-z, --sniff            sniff alike
-Z, --drone-str        *drone String
*: options with '*' require an argument following them

address ranges are cidr like 1.2.3.4/8 for all of 1.?.?.?
if you omit the cidr mask then /32 is implied
port ranges are like 1-4096 with 53 only scanning one port, a for all 65k and p for 1-1024
example: unicornscan -i eth1 -Ir 160 -E 192.168.1.0/24:1-4000 gateway:a
root@kali2019:~#
```

# PWN DEFEND

## Module 3

Mass Scan

# MASSCAN

When you are scanning large target ranges, nmap sometimes won't cut it!

```
root@kali2019:~# masscan --help
MASSCAN is a fast port scanner. The primary input parameters are the
IP addresses/ranges you want to scan, and the port numbers. An example
is the following, which scans the 10.x.x.x network for web servers:
  masscan 10.0.0.0/8 -p80
The program auto-detects network interface/adapter settings. If this
fails, you'll have to set these manually. The following is an
example of all the parameters that are needed:
  --adapter-ip 192.168.10.123
  --adapter-mac 00-11-22-33-44-55
  --router-mac 66-55-44-33-22-11
Parameters can be set either via the command-line or config-file. The
names are the same for both. Thus, the above adapter settings would
appear as follows in a configuration file:
  adapter-ip = 192.168.10.123
  adapter-mac = 00-11-22-33-44-55
  router-mac = 66-55-44-33-22-11
All single-dash parameters have a spelled out double-dash equivalent,
so '-p80' is the same as '--ports 80' (or 'ports = 80' in config file).
To use the config file, type:
  masscan -c <filename>
To generate a config-file from the current settings, use the --echo
option. This stops the program from actually running, and just echoes
the current configuration instead. This is a useful way to generate
your first config file, or see a list of parameters you didn't know
about. I suggest you try it now:
  masscan -p1234 --echo
```



# Simple Masscan example to hunt for RDP

When you are scanning a network with masscan it's a good idea to be specific about what you are looking for.

In this example we are going to scan a 24-bit mask on the 192.168.2.0 network for TCP port 3389

*masscan 192.168.2.0/24 -p3389*

```
root@kali2019:~# masscan 192.168.2.0/24 -p3389
Starting masscan 1.0.4 (http://bit.ly/14GZzcT) at 2019-02-21 09:45:30 GMT
-- forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 256 hosts [1 port/host]
Discovered open port 3389/tcp on 192.168.2.13
```

# PWIN DEFEND

## Hacking 101

Course 1 Review

# Exercises

## Exercise A

1. Practise port scanning with NMAP, UNICORNSCAN and MASSCAN
2. Try out different options and see how you can get different information such as OS version and service versions
3. See the difference in speed between NMAP and UNICORNSCAN when doing full tcp and udp scans
4. Practise turning the firewall on/off on Host B
5. Review packet data with Wireshark to see how the connections looks at a packet level

# Review

In this short course we covered:

TCP/IP Fundamentals and the 3 way-handshake and 4-step FIN

PORT Scanning using:

- NMAP
- UNICORNSCAN
- MASSCAN

# Course 1 Complete

We hope this short course content was useful and gave a decent insight into enumeration and port scanning with common tools. The subject is huge and this is just a taster.

Go and read all the thins and practise in safe environments, either internally or on great platforms like hack the box etc.

Remember to operate within your local laws! Use hacking as a force for good! #hack4good

I truly hope this was useful for you, please send comments/feedback/suggestions back to UK\_Daniel\_Card on twitter!

# PWN DEFEND

Created by

***Xservus***  
***Cyber Security***

<https://www.xservus.com>

Thank you!

#pwnDefend