Unicornscan is an asynchronous network stimulus delivery/response recording tool. Meaning it sends out broken/unorganized/fragmented packets (without a regular pattern unlike other port scanning tools) to a host and waits for the target's response.

After getting the response the TTL value is calculated for each port and thereby identifying the operating system. For eg, if the ttl=128, the operating system is Windows and so on.

Pentesters use this tool when regular port scanning doesn't work as the target might have enabled port scanning detection or has enabled IDS/IPS or honeypots. One cool feature of unicornscan is that it uses different threads to send out packets & to receive them, unlike other port scanners.

Note: This tool is not available by default in Kali Sana

Options

```
Syntax: Unicornscan <options> <target>
-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
rsts...)
-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, --covertness     *covertness 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, --safefile        *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
```

Lab 1: Scan a host for services & OS(TTL)

In this lab, we scan a host with IP address 192.168.1.250 for open ports. Also by doing so, we get the TTL value of corresponding ports and thereby we can identify the operating system

```
Command: unicornscan 192.168.1.250 -Iv
```

```
root@kali:~# unicornscan 192.168.1.250 -Iv
```

Basic Scan

```
3,941,946,992-995,1001,1023-1030,1080,1210,1214,1234,1241,1334,1349,1352,1423-14
25,1433,1434,1524,1525,1645,1646,1649,1701,1718,1719,1720,1723,1755,1812,1813,20
48-2050,2101-2104,2140,2150,2233,2323,2345,2401,2430,2431,2432,2433,2583,2628,27
76,2777,2988,2989,3050,3130,3150,3232,3306,3389,3456,3493,3542-3545,3632,3690,38
01,4000,4400,4321,4567,4899,5002,5136-5139,5150,5151,5222,5269,5308,5354,5355,54
22-5425,5432,5503,5555,5556,5678,6000-6007,6346,6347,6543,6544,6789,6838,6666-66
70,7000-7009,7028,7100,7983,8079-8082,8088,8787,8879,9090,9101-9103,9325,9359,10
000,10026,10027,10067,10080,10081,10167,10498,11201,15345,17001-17003,18753,2001
1,20012,21554,22273,26274,27374,27444,27573,31335-31338,31787,31789,31790,31791,
32668,32767-32780,33390,47262,49301,54320,54321,57341,58008,58009,58666,59211,60
000,60006,61000,61348,61466,61603,63485,63808,63809,64429,65000,65506,65530-6553
5' pps 300
using interface(s) eth0
scaning 1.00e+00 total hosts with 3.38e+02 total packets, should take a little l
onger than 8 Seconds
TCP open 192.168.1.250:22  ttl 64
TCP open 192.168.1.250:445  ttl 64
TCP open 192.168.1.250:139  ttl 64
sender statistics 294.4 pps with 338 packets sent total
listener statistics 676 packets recieved 0 packets droped and 0 interface drops
TCP open                     ssh[   22]         from 192.168.1.250  ttl 64
TCP open              netbios-ssn[  139]         from 192.168.1.250  ttl 64
TCP open            microsoft-ds[  445]         from 192.168.1.250  ttl 64
```

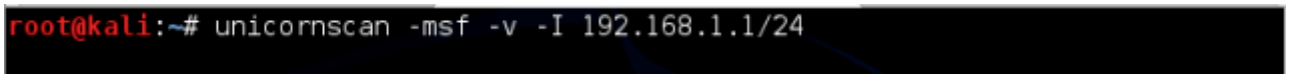
Results

Lab 2: Perform a TCP SYN Scan on a whole network

In this lab, we perform a TCP syn scan on a whole network range of 192.168.1.1/24. By doing so we have multiple benefits. All the live hosts will be visible to us along with the services/ports open & TTL values.

Stealth syn scan is a method by which packets with syn flags are sent to a port of a target host. If the port is open, the reply from the target will be a packet with SYN/ACK flag else a packet with RST flag. Thus the attacker can identify what all services are running on the target host.

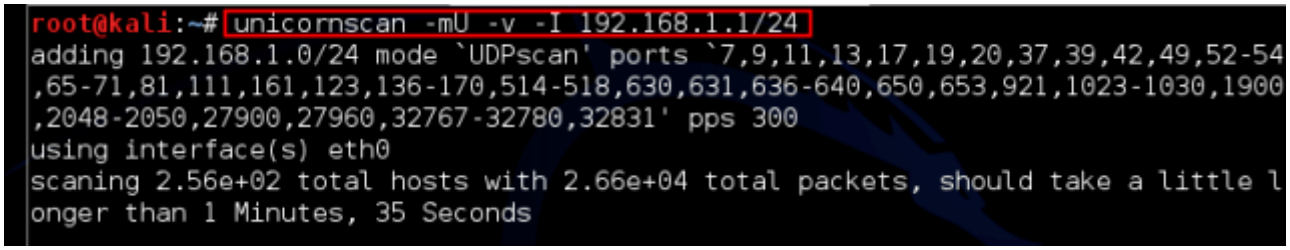
```
Command : unicornscan -msf -v -I 192.168.1.1/24
```



Lab 3: Perform a UDP scan on the whole network

In this lab, we perform a UDP scan on a whole network range of 192.168.1.1/24. By doing so we get to know all UDP services running on a network.

```
Command: unicornscan -mU -v -I 192.168.1.1/24
```



UDP Scan

While performing scans with unicornscan, turn ON Wireshark also to view the packets going out. You can see the pattern which unicornscan sends out. Try it & Share this tutorial.