Scanning Tools

The goal of the scanning phase is to learn more information about the target environment and discover openings by interacting with that target environment. This paper will look at some of the most useful scanning tools freely available today and how to best use them. During this process we'll perform a number of scans.

Scan Types

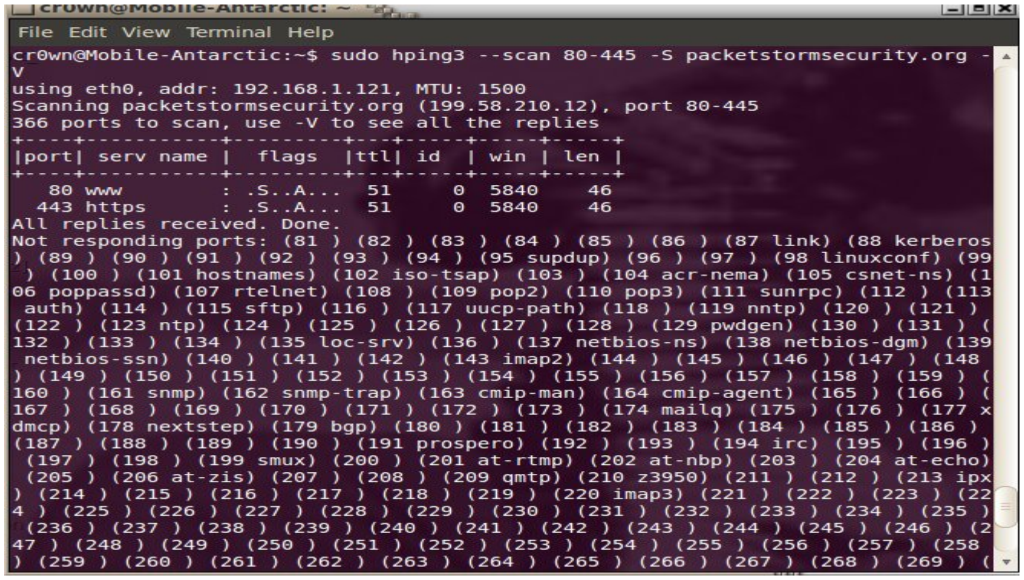
- ***Network sweeping*** - Basic technique used to determine which of a range of IP addresses map to live hosts.
- ***Network tracing* –** A facility for tracing the route of a computer that is connected to the Internet.
- ***Port scanning* –** software application designed to probe a network host for open ports.
- ***OS fingerprinting* –** analysis of the TCP/IP stack to determine target operating system.
- ***Version scanning* –** Interacting with different ports to determine protocols they speak and possibly the version of service listening on given port.
- ***Vulnerability scanning* –** used to determine a list of potential unpatched systems, misconfiguration, etc.

While performing your scans run a sniffer to monitor your network activity. This will let you know if your scan hangs or if the connection goes down.

# tcpdump -nnX tcp and dst <target>
We are not looking to catch all the packets just to make sure the connection is up and the tool we use doesn't hang.

Hping

Figure #1:port scanning TCP port 80 to 445 on host target.com.



```
cr0wn@Mobile-Antarctic:~$ sudo hping3 --scan 80-445 -S packetstormsecurity.org -
V
using eth0, addr: 192.168.1.121, MTU: 1500
Scanning packetstormsecurity.org (199.58.210.12), port 80-445
366 ports to scan, use -V to see all the replies
+----+-----------+---------+---+-----+-----+-----+
|port| serv name |  flags  |ttl| id  | win | len |
+----+-----------+---------+---+-----+-----+-----+
   80 www        : .S..A...  51      0  5840    46
  443 https      : .S..A...  51      0  5840    46
All replies received. Done.
Not responding ports: (81 ) (82 ) (83 ) (84 ) (85 ) (86 ) (87 link) (88 kerberos
) (89 ) (90 ) (91 ) (92 ) (93 ) (94 ) (95 supdup) (96 ) (97 ) (98 linuxconf) (99
 ) (100 ) (101 hostnames) (102 iso-tsap) (103 ) (104 acr-nema) (105 csnet-ns) (1
06 poppassd) (107 rtelnet) (108 ) (109 pop2) (110 pop3) (111 sunrpc) (112 ) (113
 auth) (114 ) (115 sftp) (116 ) (117 uucp-path) (118 ) (119 nntp) (120 ) (121 )
(122 ) (123 ntp) (124 ) (125 ) (126 ) (127 ) (128 ) (129 pwdgen) (130 ) (131 ) (
132 ) (133 ) (134 ) (135 loc-srv) (136 ) (137 netbios-ns) (138 netbios-dgm) (139
 netbios-ssn) (140 ) (141 ) (142 ) (143 imap2) (144 ) (145 ) (146 ) (147 ) (148
) (149 ) (150 ) (151 ) (152 ) (153 ) (154 ) (155 ) (156 ) (157 ) (158 ) (159 ) (
160 ) (161 snmp) (162 snmp-trap) (163 cmip-man) (164 cmip-agent) (165 ) (166 ) (
167 ) (168 ) (169 ) (170 ) (171 ) (172 ) (173 ) (174 mailq) (175 ) (176 ) (177 x
dmcp) (178 nextstep) (179 bgp) (180 ) (181 ) (182 ) (183 ) (184 ) (185 ) (186 )
(187 ) (188 ) (189 ) (190 ) (191 prospero) (192 ) (193 ) (194 irc) (195 ) (196 )
 (197 ) (198 ) (199 smux) (200 ) (201 at-rtmp) (202 at-nbp) (203 ) (204 at-echo)
 (205 ) (206 at-zis) (207 ) (208 ) (209 qmtp) (210 z3950) (211 ) (212 ) (213 ipx
) (214 ) (215 ) (216 ) (217 ) (218 ) (219 ) (220 imap3) (221 ) (222 ) (223 ) (22
4 ) (225 ) (226 ) (227 ) (228 ) (229 ) (230 ) (231 ) (232 ) (233 ) (234 ) (235 )
 (236 ) (237 ) (238 ) (239 ) (240 ) (241 ) (242 ) (243 ) (244 ) (245 ) (246 ) (2
47 ) (248 ) (249 ) (250 ) (251 ) (252 ) (253 ) (254 ) (255 ) (256 ) (257 ) (258
) (259 ) (260 ) (261 ) (262 ) (263 ) (264 ) (265 ) (266 ) (267 ) (268 ) (269 ) (
```

# hping3 –scan 80-445 -S target.com -V

In performing a network sweep of your target area consider using Hping which is a great general-purpose packet generation tool. Hping will ping a target IP address by sending TCP packets with no control bits set (SYN, ACK, FIN, RST, PSH, and URG set to zero).

# hping3 <target> -S -A -F -V -p 443

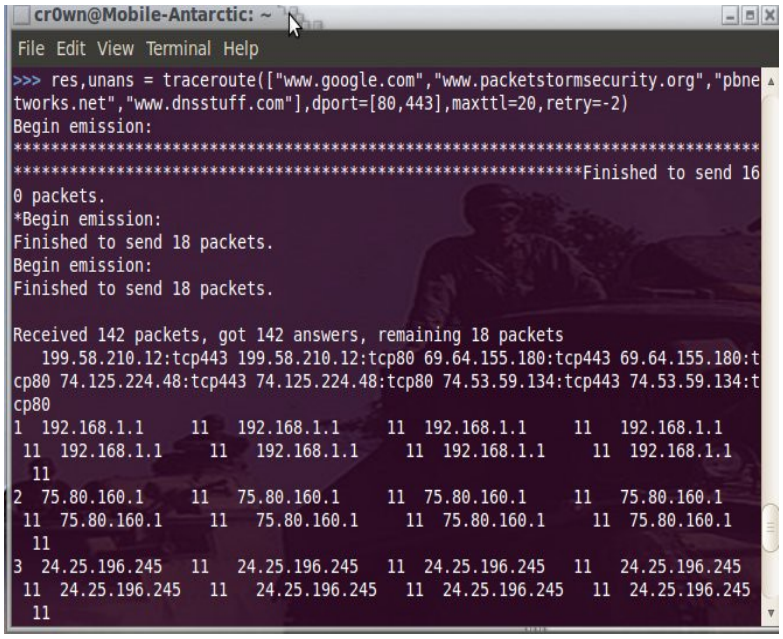
This will send TCP packets to the port 443 on host <target> with the SYN+ACK+FIN flags set.
# hping3 –rand-dest 192.168.1.x –interface eth0

This will send packets to random targets in the 192.168.1 network using interface eth0.

Scapy

Scapy is a powerful interactive packet manipulation program written in python. It can replace hping and a number of other tools such as arping, tcpdump, tethereal, and p0f. Scapy's interactive shell is run in a terminal session. Root privileges are needed to send the packets or sudo.

>>>res,unans=traceroute([“www.google.com”,”www.packetstormsecurity.org”,”pbnetworks.net”,”www.dnsstuff.com”],dport=[80,443],maxttl=20,retry=-2)
The above is all on one line. (see Figure #2)
>>> res.graph()



```
>>> res,unans = traceroute(["www.google.com","www.packetstormsecurity.org","pbne
tworks.net","www.dnsstuff.com"],dport=[80,443],maxttl=20,retry=-2)
Begin emission:
********************************************************************************
*************************************************************Finished to send 16
0 packets.
*Begin emission:
Finished to send 18 packets.
Begin emission:
Finished to send 18 packets.

Received 142 packets, got 142 answers, remaining 18 packets
   199.58.210.12:tcp443 199.58.210.12:tcp80 69.64.155.180:tcp443 69.64.155.180:t
cp80 74.125.224.48:tcp443 74.125.224.48:tcp80 74.53.59.134:tcp443 74.53.59.134:t
cp80
1  192.168.1.1     11   192.168.1.1     11  192.168.1.1     11   192.168.1.1
 11  192.168.1.1     11   192.168.1.1     11  192.168.1.1     11  192.168.1.1
  11
2  75.80.160.1     11   75.80.160.1     11  75.80.160.1     11   75.80.160.1
 11  75.80.160.1     11   75.80.160.1     11  75.80.160.1     11  75.80.160.1
  11
3  24.25.196.245   11   24.25.196.245   11  24.25.196.245   11   24.25.196.245
 11  24.25.196.245   11   24.25.196.245   11  24.25.196.245   11  24.25.196.245
  11
```

Figure #2: Scapy in Action

This will output a graph in ImageMagick that will display your traceroute. (see Figure #3) This is just a small portion of what Scapy can do for the pen-tester. I have two in depth videos on my site describing the many uses of Scapy <http://pbnetworks.net/?cmd=bbs&id=42>,

<http://pbnetworks.net/?cmd=bbs&id=43> they are about 10 and 13 min in length.

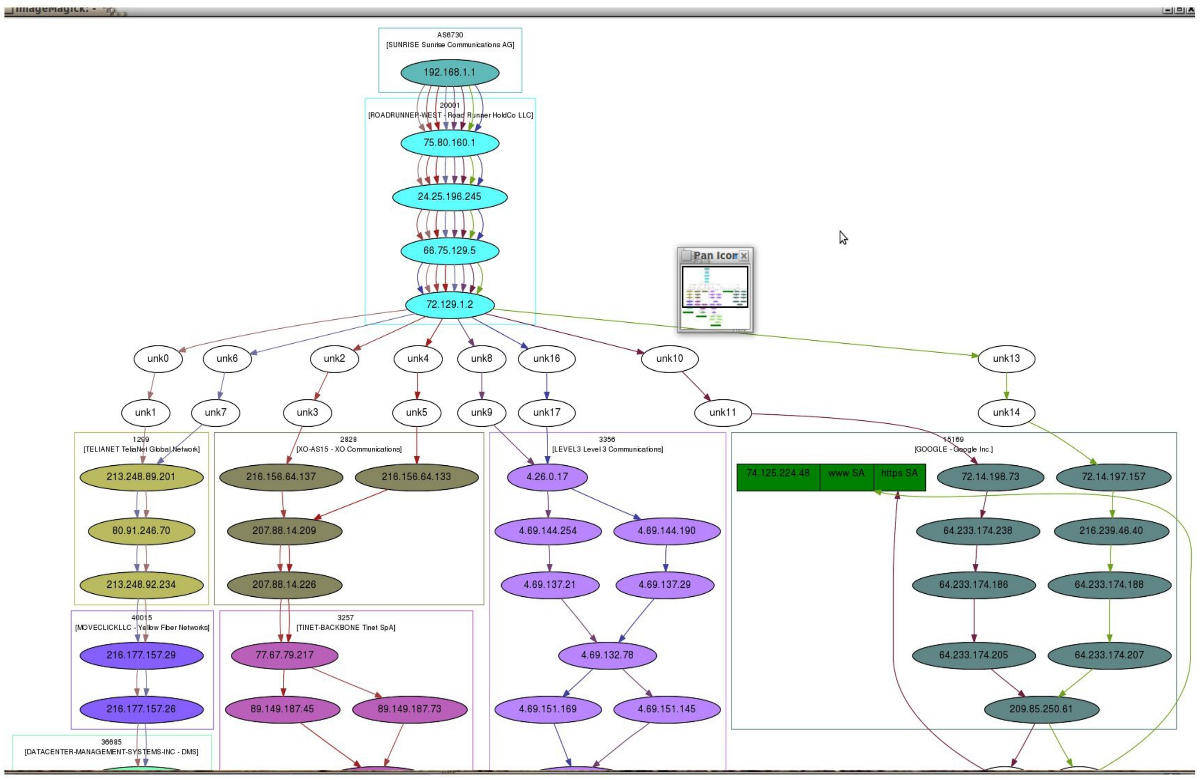


Figure #3 :Syn Scanning a target

Scapy can provide a valuable scanning tool as well. We will use scapy and its ability to SYN scan a target and look for open, closed, or filtered ports. We will scan an internal and external IP address below.

>>> sr(IP(dst="192.168.1.111")/TCP(sport=RandShort(),dport=[21,22,80,443,445],flags="S"))

This command will send a SYN scan on target 192.168.1.111 to destination ports 21,22,80,443, and 445. If we would like to scan a whole range use the () around the port numbers instead of the [ ]. Next we look at the response by issuing a request summary

>>> ans,unans =_ and *>>> ans.summary()*

We can display only the information that we are interested in by using a simple loop

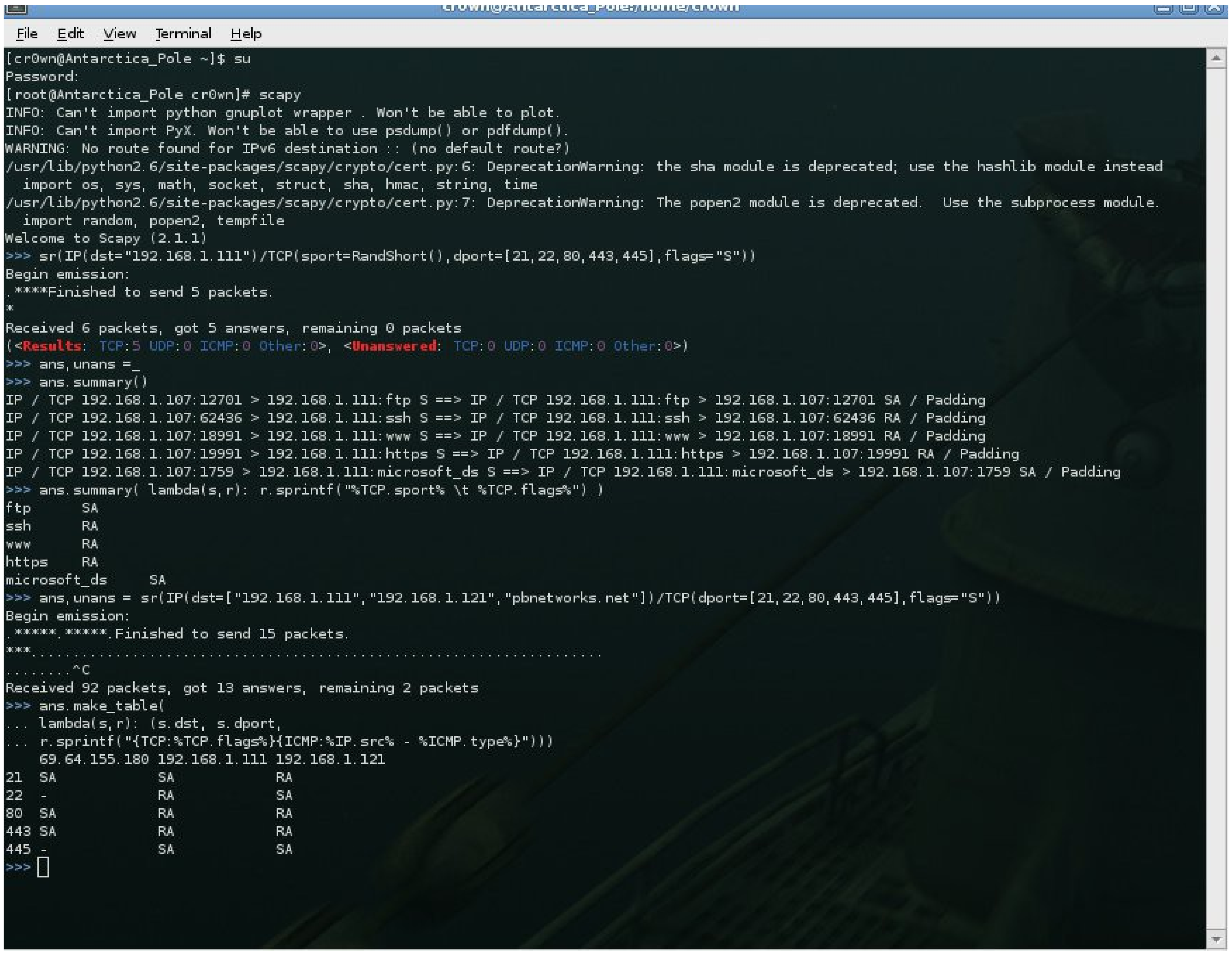
>>> ans.summary(lambda(s,r): r.sprintf("%TCP.sport% \t %TCP.flags%"))

Still a better table can be built with the make_table() function. Fist lets add some targets to scan.

>>> ans,unans = sr(IP(dst=["192.168.1.111","192.168.1.121","pbnetworks.net"])/TCP(dport=[21,22,80,443,445],flags="S"))

>>> ans.make_table(
…   lambda(s,r): (s.dst, s.dport,
…   r.sprintf("{TCP:%TCP.flags%}{ICMP:%IP.src% - %ICMP.type%}")))

The output of the above can be seen in Figure #4



```
[cr0wn@Antarctica_Pole ~]$ su
Password:
[root@Antarctica_Pole cr0wn]# scapy
INFO: Can't import python gnuplot wrapper . Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
WARNING: No route found for IPv6 destination :: (no default route?)
/usr/lib/python2.6/site-packages/scapy/crypto/cert.py:6: DeprecationWarning: the sha module is deprecated; use the hashlib module instead
  import os, sys, math, socket, struct, sha, hmac, string, time
/usr/lib/python2.6/site-packages/scapy/crypto/cert.py:7: DeprecationWarning: The popen2 module is deprecated.  Use the subprocess module.
  import random, popen2, tempfile
Welcome to Scapy (2.1.1)
>>> sr(IP(dst="192.168.1.111")/TCP(sport=RandShort(),dport=[21,22,80,443,445],flags="S"))
Begin emission:
.****Finished to send 5 packets.
*
Received 6 packets, got 5 answers, remaining 0 packets
(<Results: TCP:5 UDP:0 ICMP:0 Other:0>, <Unanswered: TCP:0 UDP:0 ICMP:0 Other:0>)
>>> ans,unans =_
>>> ans.summary()
IP / TCP 192.168.1.107:12701 > 192.168.1.111:ftp S ==> IP / TCP 192.168.1.111:ftp > 192.168.1.107:12701 SA / Padding
IP / TCP 192.168.1.107:62436 > 192.168.1.111:ssh S ==> IP / TCP 192.168.1.111:ssh > 192.168.1.107:62436 RA / Padding
IP / TCP 192.168.1.107:18991 > 192.168.1.111:www S ==> IP / TCP 192.168.1.111:www > 192.168.1.107:18991 RA / Padding
IP / TCP 192.168.1.107:19991 > 192.168.1.111:https S ==> IP / TCP 192.168.1.111:https > 192.168.1.107:19991 RA / Padding
IP / TCP 192.168.1.107:1759 > 192.168.1.111:microsoft_ds S ==> IP / TCP 192.168.1.111:microsoft_ds > 192.168.1.107:1759 SA / Padding
>>> ans.summary( lambda(s,r): r.sprintf("%TCP.sport% \t %TCP.flags%") )
ftp      SA
ssh      RA
www      RA
https    RA
microsoft_ds     SA
>>> ans,unans = sr(IP(dst=["192.168.1.111","192.168.1.121","pbnetworks.net"])/TCP(dport=[21,22,80,443,445],flags="S"))
Begin emission:
.*****.*****.Finished to send 15 packets.
***.....................................................................
........^C
Received 92 packets, got 13 answers, remaining 2 packets
>>> ans.make_table(
... lambda(s,r): (s.dst, s.dport,
... r.sprintf("{TCP:%TCP.flags%}{ICMP:%IP.src% - %ICMP.type%}")))
    69.64.155.180 192.168.1.111 192.168.1.121
21  SA            SA            RA
22  -             RA            SA
80  SA            RA            RA
443 SA            RA            RA
445 -             SA            SA
>>> 
```

Figure #4: showing open ports

Taking a look at the output of the above SYN scan we see that 69.64.155.180 has port 21 open, port 22 filtered or closed, ports 80 & 443 open, and port 445 filtered or closed. IP address 192.168.1.111 has port 21 open, ports 22,80, & 443 closed, and port 445 open. IP address 192.168.1.121 has port 21 closed, port 22 open, ports 80 & 443 closed, and port 445 open. The same scan can be done with ACK using *flags="A")) &* Xmas scan using *flags="FPU"))*.

TCP port scanning sends a TCP SYN on each port and waits for a SYN-ACK or a RST or an ICMP error.

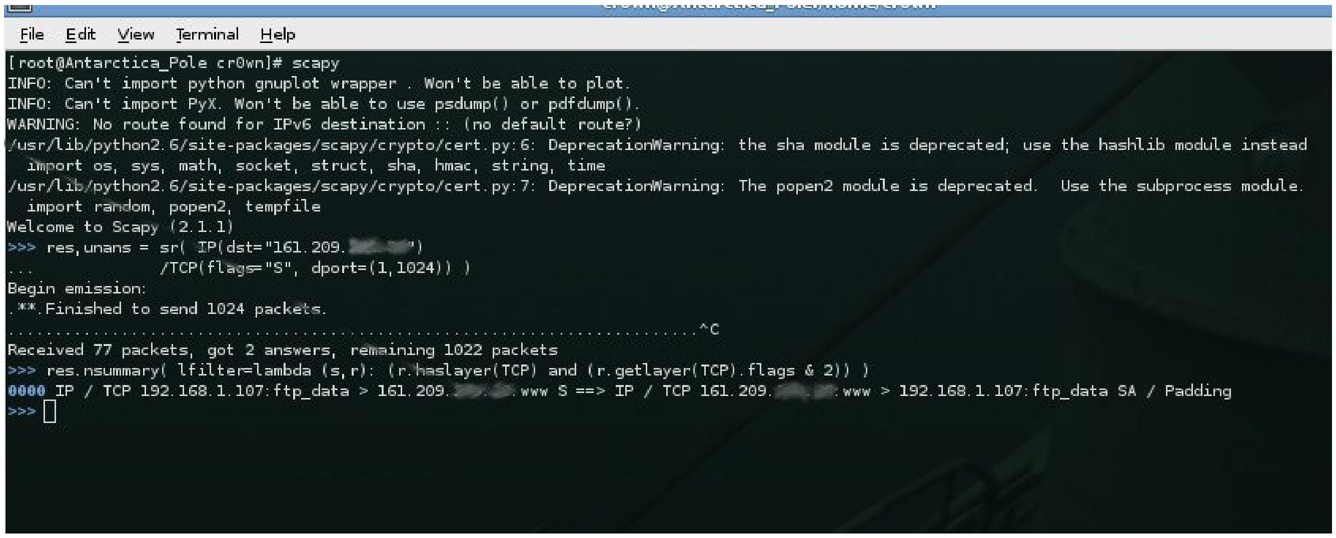
>>> res,unans = sr(IP(dst="target")
…   /TCP(flags="S", dport=(1,1024)))

This will send out a TCP SYN on ports 1-1024. To view the results use this command:

>>> res.nsummary(lfilter=lambda (s,r): (r.haslayer(TCP) and (r.getlayer(TCP).flags & 2)))

The output of the above can be seen in Figure # 5

Figure #5: TCP SYN on ports 1-1024



```
[root@Antarctica_Pole cr0wn]# scapy
INFO: Can't import python gnuplot wrapper . Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
WARNING: No route found for IPv6 destination :: (no default route?)
/usr/lib/python2.6/site-packages/scapy/crypto/cert.py:6: DeprecationWarning: the sha module is deprecated; use the hashlib module instead
  import os, sys, math, socket, struct, sha, hmac, string, time
/usr/lib/python2.6/site-packages/scapy/crypto/cert.py:7: DeprecationWarning: The popen2 module is deprecated.  Use the subprocess module.
  import random, popen2, tempfile
Welcome to Scapy (2.1.1)
>>> res,unans = sr( IP(dst="161.209.[redacted]")
...              /TCP(flags="S", dport=(1,1024)) )
Begin emission:
.**.Finished to send 1024 packets.
...........................................................................^C
Received 77 packets, got 2 answers, remaining 1022 packets
>>> res.nsummary( lfilter=lambda (s,r): (r.haslayer(TCP) and (r.getlayer(TCP).flags & 2)) )
0000 IP / TCP 192.168.1.107:ftp_data > 161.209.[redacted].www S ==> IP / TCP 161.209.[redacted]:www > 192.168.1.107:ftp_data SA / Padding
>>>
```

UDP Ping

To use UDP ping to produce ICMP port unreachable errors from a live host use the following cmd:

>>> ans,unans=sr(IP(dst=”192.168.1.121”)/UDP(dport=0))
Begin emission:
.Finished to send 1 packets.
***
Received 2 packets, got 1 answers, remaining 0 packets
>>> ans.summary(lambda(s,r) : r.sprintf(“%IP.src% is alive”))
192.168.1.121 is alive

Scapy is very good at two things sending packets and receiving answers. You can define your set of packets, it will send them, get answers back, match requests with answers, and list the unmatched packets. Scapy is not designed for fast throughput. It is written in Python, has many layers of abstraction (memory intensive). Do not expect a packet rate higher than 6 Mbs per second. You can easily design something that sniffs, mangles, and sends and this is exactly what is needed for the pentester.

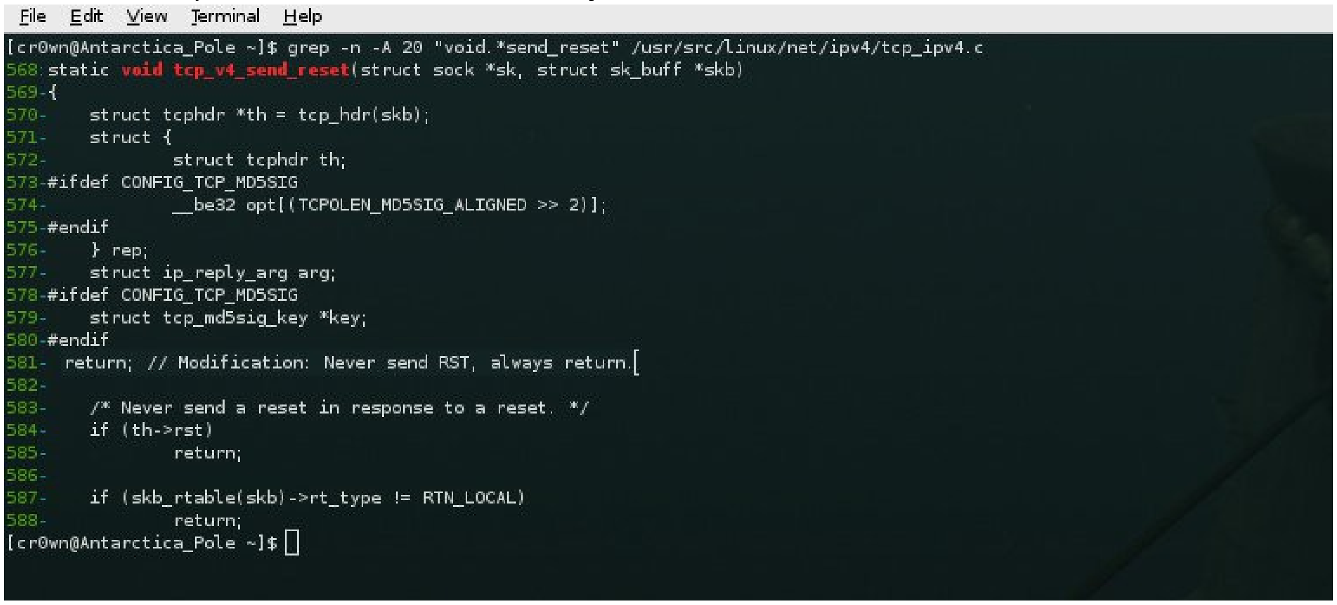
Defense

It is possible to prevent port scans before they happen. In the case of FIN, Null, and X-mas scans can be prevented by simple kernel modification. If the kernel never sends a reset packet then the scan will turn up nothing. Lets take a look at the kernel code responsible for sending reset packets:

*$ grep -n -A -20 “void.*send_reset” /usr/src/linux/net/ipv4/tcp_ipv4.c*

Add a return command at line 581 (see Figure #6 may be different line on your system) with this:

return; // Modification: Never send RST, always return



```
[cr0wn@Antarctica_Pole ~]$ grep -n -A 20 "void.*send_reset" /usr/src/linux/net/ipv4/tcp_ipv4.c
568:static void tcp_v4_send_reset(struct sock *sk, struct sk_buff *skb)
569-{
570-    struct tcphdr *th = tcp_hdr(skb);
571-    struct {
572-            struct tcphdr th;
573-#ifdef CONFIG_TCP_MD5SIG
574-            __be32 opt[(TCPOLEN_MD5SIG_ALIGNED >> 2)];
575-#endif
576-    } rep;
577-    struct ip_reply_arg arg;
578-#ifdef CONFIG_TCP_MD5SIG
579-    struct tcp_md5sig_key *key;
580-#endif
581-  return; // Modification: Never send RST, always return.
582-
583-    /* Never send a reset in response to a reset. */
584-    if (th->rst)
585-            return;
586-
587-    if (skb_rtable(skb)->rt_type != RTN_LOCAL)
588-            return;
[cr0wn@Antarctica_Pole ~]$
```

Figure #6:changing the code responsible for sending reset packets.

Nmap

Nmap is a free open source utility for network security auditing. This tool is useful once we've got a list of open ports and need to determine which services are using those ports. Finding these services running on ports we can use the Nmap version scanning functionality.

# nmap -n -O -sT -sV -p 1-1024 <target>

When using the -sV option you invoke the version scanning functionality. This will allow you to find services that are running on non-standard ports such as a web server running on TCP 90 or sshd on TCP 4444.

$ sudo nmap –spoof-mac Apple –traceroute –data-length 9 \ -f -D <victim>,RND:5,ME -v -n -O -sS -sV -p T:1-1024 \ --randomize-hosts <target>

Now this is an interesting port scan we are spoofing our mac with the *–spoof-mac* option to appear to be a Apple OS. We complete a traceroute with the *–traceroute* and append the data length to 9 with *–data-length 9* which will fill an entire TCP packet. We follow this with the *-f* which will cut up the data packets into 8 bit segments. The *-D* will allow us to add a decoy which is labeled <victim>. The *RND:5* tells nmap to come up with 5 IP random IP address, *ME* tells nmap to put our IP in as the 7th IP address which in some IDS's will never be logged. Followed by *-v* to increase verbosity level *-n* no DNS resolution, *-O* OS detection, *-sS* TCP SYN stealth scan, *-sV* version scan, *-p T:1-1024* specifies to only scan ports 1-1024 TCP and then randomize the hosts *–randomize-hosts*. While this will not completely provide anonymity it throws enough random IP at the host you are scanning to create a lot of work on the hosts intrusion detection team. Using the -f fragment packet option causes the requested scan to use tiny fragmented IP packets which makes it harder for packet filters, IDS, and other annoyances to detect what you are doing. The -D decoy scan which makes it appear to the remote host you specify as decoys are scanning the target network too. When you put ME in the list of decoys to represent the position for your real IP address in the sixth position or later, some common port scan detectors are unlikely to show your IP address at all. The –randomize-hosts tells nmap to shuffle each group of up to 16384 hosts before it scans them making the scan less obvious to various network monitoring systems.

Nmap Scriping Engine Scripts

Nmaps scripts are written in the Lua scripting language is fast, flexible, and free, with a small interpreter that works across platforms and is easily embedded inside of other applications. To invoke the nmap scripting engine, a user would use the -sC option to run all scripts in the 'default' category or with the –script optin to choose specific scripts.

```
# nmap -sC <target> -p 1-1024
```

Script categories consist of safe, intrusive, auth, malware, version, discovery, vulnerability, and default. Scripts are located in /usr/share/nmap/scripts/script.db. The following example is using the NSE script on a targets web applications:

```
# nmap –script “http-*” <target> -p 80,443
```

This will use all the NSE scripts that apply to http against the target box on ports 80 and 443.
To run a specific script use the --script parameter followed by the name of the script or a group of scripts: *# nmap –script Intrusive <target>*

UDP Scan Types

UDP stands for User Datagram Protocol, very basic and lightweight, with few safeguards built into it. UDP uses datagram sockets on the transport layer (4), is one-way only and unreliable, just a basic method for sending data from one point to another. Nmap and unicornscan are tools that use UDP protocols. There are two types: empty packet scans and protocol data scans.

Empty packet scans – send UDP packets without any data to a port and wait to see whether a result is returned. If no response is seen then the port is considered open or filtered. If port unreachable is returned it can be assumed that the port is closed.

Protocol data scans – sending valid application protocol data in UDP packets to ports to see whether an application responds. Since this involves talking to the application it is more likely to be logged.

For nmap to perform an empty packet scan use the -sU flag:

```
# nmap -sU <target>
```

Unicornscan supports protocol data scans only, use the -mU flag to perform this scan:

```
# unicornscan -mU <target>
```

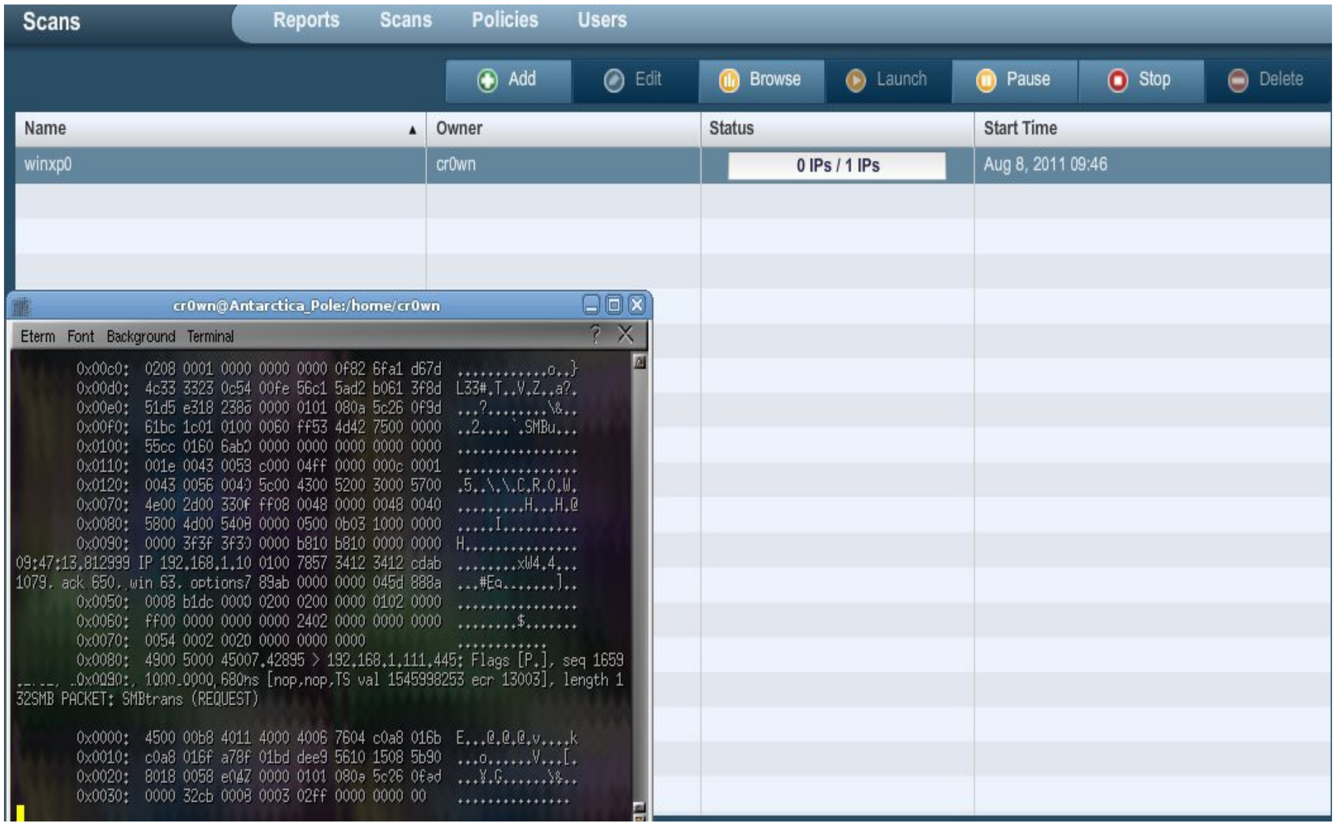
Nmap can use the applicaiotn fingerprint to perform protocol data scan:

# nmap -sU -sV <target>

Mixing UDP and application fingerprinting scans in Nmap can be very slow, if used limit the ports to be scanned.

Nessus Vulnerability Scanner

Figure #7: Nessus Scan



Name	Owner	Status	Start Time
winxp0	cr0wn	0 IPs / 1 IPs	Aug 8, 2011 09:46

```
        0x00c0:  0208 0001 0000 0000 0000 0f82 6fa1 d67d  ............o..}
        0x00d0:  4c33 3323 0c54 00fe 56c1 5ad2 b061 3f8d  L33#.T..V.Z..a?.
        0x00e0:  51d5 e318 2386 0000 0101 080a 5c26 0f9d  ...?.........\&..
        0x00f0:  61bc 1c01 0100 0060 ff53 4d42 7500 0000  ..2.....`.SMBu...
        0x0100:  55cc 0160 6ab0 0000 0000 0000 0000 0000  ................
        0x0110:  001e 0043 0053 c000 04ff 0000 000c 0001  ................
        0x0120:  0043 0056 0040 5c00 4300 5200 3000 5700  .5..\.\.C.R.0.W.
        0x0070:  4e00 2d00 330f ff08 0048 0000 0048 0040  ..........H...H.@
        0x0080:  5800 4d00 5408 0000 0500 0b03 1000 0000  .....I..........
        0x0090:  0000 3f3f 3f30 0000 b810 b810 0000 0000  H...............
09:47:13.812999 IP 192.168.1.10 0100 7857 3412 3412 cdab  .........xW4.4...
1079. ack 650. win 63. options7 89ab 0000 0000 045d 888a  ...#Eq........]..
        0x0050:  0008 b1dc 0000 0200 0200 0000 0102 0000  ................
        0x0060:  ff00 0000 0000 0000 2402 0000 0000 0000  .........$.......
        0x0070:  0054 0002 0020 0000 0000 0000            .............
        0x0080:  4900 5000 45007.42895 > 192.168.1.111.445: Flags [P.], seq 1659
  ..0x0090:  1000 0000 680ns [nop,nop,TS val 1545998253 ecr 13003], length 1
32SMB PACKET: SMBtrans (REQUEST)

        0x0000:  4500 00b8 4011 4000 4006 7604 c0a8 016b  E...@.@.@.v....k
        0x0010:  c0a8 016f a78f 01bd dee9 5610 1508 5b90  ...o.......V...[.
        0x0020:  8018 0058 e047 0000 0101 080a 5c26 0fad  ...X.G.......\&..
        0x0030:  0000 32cb 0008 0003 02ff 0000 0000 00    ...............
```

The Nessus Vulnerability Scanner is distributed by Tenable Network Security. It is available for download in Nessus for Business and Nessus for Home. I will not go into the installation or integration with Metasploit as I wrote another article on the subject for ClubHack earlier this year. I will go over the interface and demo a couple of scans on windows boxes looking for vulnerabilities.
First lets update the version of nessus by issuing the following command:

/opt/nessus/sbin/nessus-update-plugins

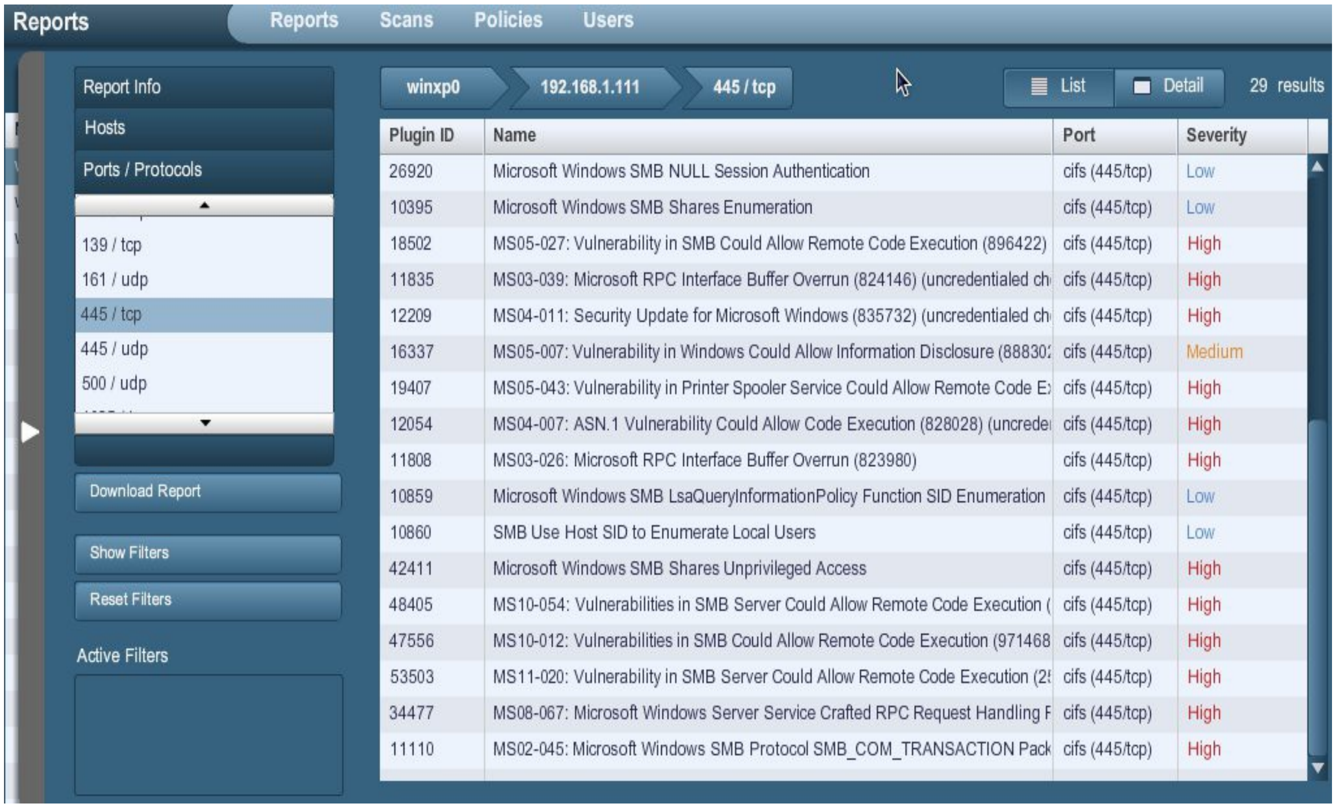
then start nessusd

# ./nessusd

Next we will open up a web-browser to *https://localhost:8834* and login. There you can create policies, review reports, launch scans, etc. To begin a scan choose add and specify a target in this case its 192.168.1.111 and select a type of scan and this one we select Internal Network Scan. The scan is completed in short order notice I have tcpdump running when I launch the scan. (see Figure #7)

Nessus results include an estimate of the risk level associated with the finding (High, Medium, or Low) along with a brief description of each flaw along with recommendations for remedy (see Figure #8)

Figure #8: Nessus Scan Report.



Plugin ID	Name	Port	Severity
26920	Microsoft Windows SMB NULL Session Authentication	cifs (445/tcp)	Low
10395	Microsoft Windows SMB Shares Enumeration	cifs (445/tcp)	Low
18502	MS05-027: Vulnerability in SMB Could Allow Remote Code Execution (896422)	cifs (445/tcp)	High
11835	MS03-039: Microsoft RPC Interface Buffer Overrun (824146) (uncredentialed ch	cifs (445/tcp)	High
12209	MS04-011: Security Update for Microsoft Windows (835732) (uncredentialed ch	cifs (445/tcp)	High
16337	MS05-007: Vulnerability in Windows Could Allow Information Disclosure (88830	cifs (445/tcp)	Medium
19407	MS05-043: Vulnerability in Printer Spooler Service Could Allow Remote Code E	cifs (445/tcp)	High
12054	MS04-007: ASN.1 Vulnerability Could Allow Code Execution (828028) (uncreder	cifs (445/tcp)	High
11808	MS03-026: Microsoft RPC Interface Buffer Overrun (823980)	cifs (445/tcp)	High
10859	Microsoft Windows SMB LsaQueryInformationPolicy Function SID Enumeration	cifs (445/tcp)	Low
10860	SMB Use Host SID to Enumerate Local Users	cifs (445/tcp)	Low
42411	Microsoft Windows SMB Shares Unprivileged Access	cifs (445/tcp)	High
48405	MS10-054: Vulnerabilities in SMB Server Could Allow Remote Code Execution (	cifs (445/tcp)	High
47556	MS10-012: Vulnerabilities in SMB Could Allow Remote Code Execution (971468	cifs (445/tcp)	High
53503	MS11-020: Vulnerability in SMB Server Could Allow Remote Code Execution (2	cifs (445/tcp)	High
34477	MS08-067: Microsoft Windows Server Service Crafted RPC Request Handling F	cifs (445/tcp)	High
11110	MS02-045: Microsoft Windows SMB Protocol SMB_COM_TRANSACTION Pack	cifs (445/tcp)	High

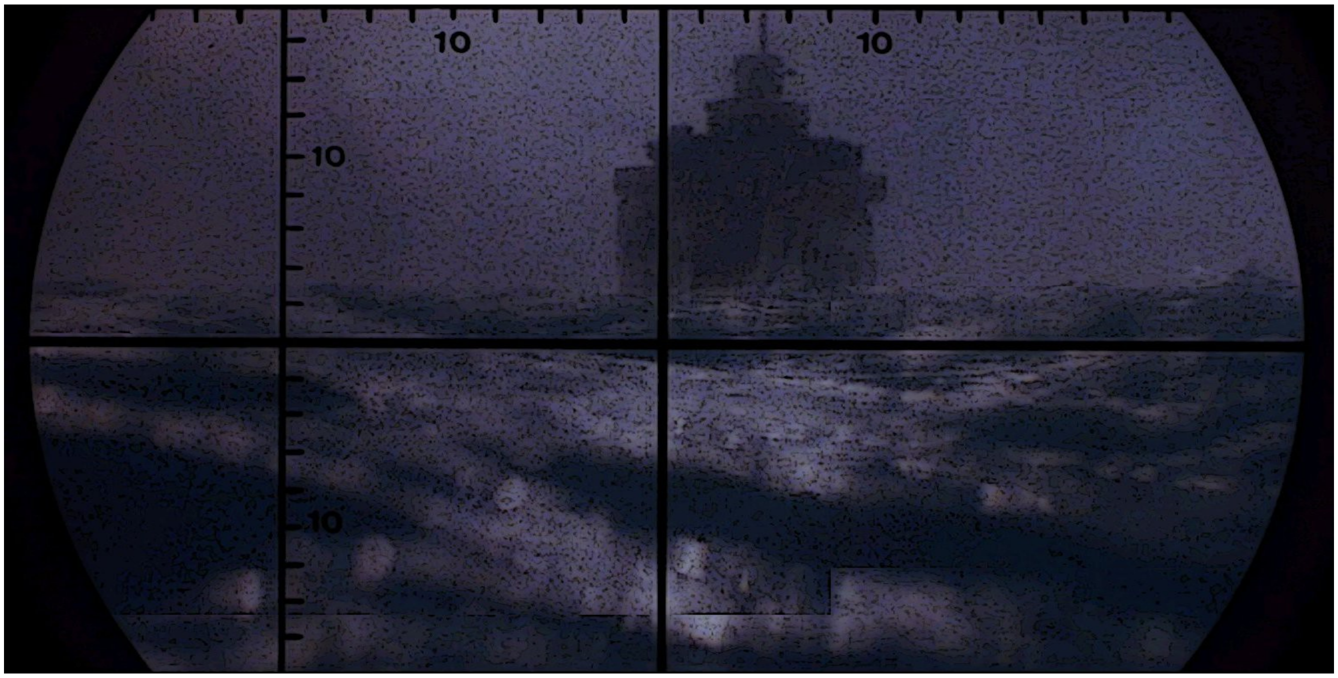
Conclusion

The scanning phase a penetration tester will acquire useful information about the target environment that will be critical in the following stages where we use it to exploit our target. I have analyzed some tools used in determining many things about the target environment such as open ports, operating system types, and vulnerabilities.

About the Author

David J. Dodd is currently in the United States and holds a current 'Secret' DoD Clearance and is available for consulting on various Information Assurance projects. A former U.S. Marine with Avionics background in Electronic Countermeasures Systems. David has given talks at the San Diego Regional Security Conference and SDISSA, is a member of InfraGard, and contributes to Secure our eCity http://securingourecity.org. He works for pbnetworks Inc. http://pbnetworks.net a small service disabled veteran owned business located in San Diego, CA and can be contacted by emailing: dave@pbnetworks.net.

Let pbnetworks get your pen test on target
How secure is your network?



Visit us and learn how
http://pbnetworks.net