Intrusion Detection Practical Assignment
for SANS Security DC 2000
by Joseph R. Rach

Introduction

This document contains three assignments: Network Detects, Evaluation of an Attack, and an "Analyze This" Scenario. The five detects and evaluation where conducted on a network designed specifically for analyzing intrusion attempts. To help accomplish this, network countermeasures were purposely set low. We will see comments and suggestions about this in each detect. The format for the detect analysis is specified in the assignment documentation. The data for the "Analyze This" Scenario was provided with the assignment documentation.

The intrusion detection system used was Snort with a generic set of rules. Tcpdump data was also collected. Both destination and source IP address were sanitized for anonymity (or security reasons) with the following rule:

All attacker addresses -> SCANNER.OTHER.NET
All internal addresses -> ***.MY.NET

Readers are assumed to have at minimum a basic understanding of the Internet Protocol suite.

Table of Contents:
- Introduction
- Network Detect 1
- Network Detect 2
- Network Detect 3
- Network Detect 4
- Network Detect 5
- Attack Evaluation
- "Analyze This" Scenario

Detect 1 - A DNS Version Scan and Zone Transfer

The following is Snort output data:

```
[**] IDS277 - NAMED Iquery Probe [**]
UDP TTL:64 TO8:0x0 ID:48361 Len: 35

[**] MISC-DNS-version-query [**]
UDP TTL:64 TO8:0x0 ID:48362 Len: 36

[**] IDS212 - MISC - DNS Zone Transfer [**]
TCP TTL:64 TO8:0x0 ID:48366 DF
*****PA* Seq: 0
x7663C408   Ack: 0
x8DADD372   Win: 0
x4470
```

The following is tcpdump output data:

```
22:26:16.869288 SCANNER.OTHER.NET.1132 > DNS_SERVER.MY.NET.12329 inv_q+ [b2&3=0x980] A? . (27) (ttl 64, id 48361)
22:26:16.875275 DNS_SERVER.MY.NET.12329 > SCANNER.OTHER.NET.1132: 12329 inv_q q: [4.3.2.1]. 1/0/0 . (42) (ttl 64, id 52055)
22:26:16.875704 SCANNER.OTHER.NET.1132 > DNS_SERVER.MY.NET: 13448+ [b2&3=0x180] (30) (ttl 64, id 48362)
22:26:17.059765 DNS_SERVER.MY.NET.13448 > SCANNER.OTHER.NET.1200: 13448* q: version.bind. 1/0/0  (63) (ttl 64, id 48363)
22:26:17.099362 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET: S 1986249733:1986249733(0) win 16384 (DF) (ttl 64, id 48364)
22:26:17.099770 DNS_SERVER.MY.NET.1986249733 > SCANNER.OTHER.NET.1200: S 2376979313:2376979313(0) ack 1986249734 win 17520 (ttl 64, id 48365)
22:26:17.100400 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET: P 1:3(2) ack 1 win 17520 (DF) (ttl 64, id 48366)
22:26:17.102376 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.1461:1461(1460) ack 30 win 17520 (ttl 64, id 48367)
22:26:17.102669 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.1461:1461(1460) ack 30 win 17520 (ttl 64, id 48368)
22:26:17.104083 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.1461:1461(1460) ack 30 win 17520 (ttl 64, id 48369)
22:26:17.104083 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.1461:1461(1460) ack 30 win 17520 (ttl 64, id 48369)
22:26:17.183542 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.1461:1461(1460) ack 30 win 17520 (ttl 64, id 48369)
22:26:17.184045 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.1461:1461(1460) ack 30 win 17520 (ttl 64, id 48369)
22:26:17.184943 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.1461:1461(1460) ack 30 win 17520 (ttl 64, id 48369)
22:26:17.185066 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.1461:1461(1460) ack 30 win 17520 (ttl 64, id 48369)
22:26:17.211787 SCANNER.OTHER.NET.1200 > DNS_SERVER.MY.NET.1461:1461(1460) ack 30 win 17520 (ttl 64, id 48369)
22:26:17.212321 DNS_SERVER.MY.NET.1461:1461(1460) ack 31 win 17520 (ttl 64, id 60072)
```

The following is syslog output data:

```
Aug 12 22:26:15 DNS_SERVER named[19779]: XX /DNS_SERVER/DNS_SERVER/-A
Aug 12 22:26:15 DNS_SERVER named[19779]: XX /DNS_SERVER/DNS_SERVER/VERSION.BIND/TEXT
Aug 12 22:26:16 DNS_SERVER named[19779]: approved AXFR from [SCANNER.OTHER.NET].1200 for 'MY.NET'
Aug 12 22:26:16 DNS_SERVER named[19779]: XX /DNS_SERVER/MY.NET/AXFR
```

© SANS Institute 2000 - 2005
Author retains full rights.
1. Source of trace:
A network designed specifically for analyzing intrusion attempts with little or no network countermeasures.

2. Detect was generated by:
Detected by Snort (The Lightweight Network Intrusion Detection System) with a full ruleset, tcpdump, and syslog.

3. Probability the source address was spoofed:
The probability is low, because the attacker wants to see the response. The DNS Zone Transfer (TCP) trace gives high confidency to the source address being the real deal.

4. Description of the attack:
The attacker is scanning to find the version of BIND running on our DNS server and requests a DNS Zone Transfer. This appears to be a reconnaissance, and could be followed up by CVE-1999-0833, 0009, 0835, 0848, 0849, and/or 0851. Additionally, BIND weaknesses are number 1 on SANS Institute Top 10.

5. Attack Mechanism:
This attack mechanism works by doing an inverse DNS query to determine the version of BIND running on the system. Given the version number a targeted remote root compromise can be launched provided a compromisable version is running. Additionally, the attacker attempted a DNS zone transfer to find hostnames and addresses in our network. This information can then be used to better target future scanning.

6. Correlations:
This particular detect is not new. Buffer overflows against DNS are well know and are considered in the top ten list (www.sans.org/topten.htm). The CVE numbers listed above are reports previously issued on the subject.

7. Evidence of active targeting:
The attacker is just starting active targeting by getting our DNS maps and determining the version of BIND we are using. We could see a buffer overflow attempt against our DNS server in the near future.

8. Severity:
Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures)
Criticality: 5 (The destination host is a core DNS server)
Lethality: 2 (This attack is acquiring information about our network)
System Countermeasures: 4 (Modern OS, all patches, additional security)
Network Countermeasures: 1 (Little to no protection from firewalls)
Severity = (5 + 2) - (4 + 1) = 2.

NOTE: Since the zone transfer was successful, we may want to increase our severity rating by 1. Also, the severity would have been greatly increased if a buffer overflow was attempted.

9. Defensive recommendation:
Recommendation is to implement a packet filter and firewall to deny all packets requesting our BIND version and Zone Transfers. Additionally, we should double check our BIND implementation to make sure it is running in a chroot( ) environment with non-root privileges (www.psionic.com/papers/dns) and disable zone transfers to the outside. Finally, we may want to review the zone map to see how much information the attacker now has about our site and verify patching and logging procedures are being followed.

10. Multiple choice test question:

Detect 2 - A rpc.statd buffer overflow attempt

The following is Snort output data:

```
[**] IDS15 - RPC - portmap-request-status [**]
08/12-22:32:27.256042 SCANNER.OTHER.NET:783 -> NFS_SERVER.MY.NET:111
```

These syslog entries suggest:

a) SCANNER.OTHER.NET successfully poisoned DNS_SERVER's cache.
b) SCANNER.OTHER.NET attempted a remote buffer overflow attack against DNS_SERVER.
c) It is normal to see a request for BIND's version before requesting and AXFR.
d) SCANNER.NET requested a zone transfer and was approved.

Answer: d
The following is tcpdump output data:

```
```

The following is syslog output data:

```
```

The following is output data from rpcinfo -p:

```
program vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100005    3   tcp    2049  nfs
100005    2   tcp    2049  nfs
100005    3   udp    2049  nfs
100005    2   udp    2049  nfs
100005    1   tcp    1023  mountd
100005    1   udp    1023  mountd
100005    3   tcp    1023  mountd
100005    3   udp    1023  mountd
```

1. Source of trace:

A network designed specifically for analyzing intrusion attempts with little or no network countermeasures.

2. Detect was generated by:

Detected by Snort (The Lightweight Network Intrusion Detection System) with a full ruleset, tcpdump, syslog, and rpcinfo.

3. Probability the source address was spoofed:

The probability is about 50/50, because the attacker used the portmapper to find the port being used to rpc.statd. This could just be a decoy and the attacker could have just gone after the "well-known" ports that rpc.statd runs on. Also, this could be a man-in-the-middle type attack (i.e. the attacker sniffs the UDP going back to a spoofed address). Since this attack is using UDP, the overflow could just be a remote command to open a hole to attack with later.

4. Description of the attack:

The attacker is attempting a remote buffer overflow on our rpc.statd daemon used for NFS. This appears to be an attempt to execute a command on our NFS server to open a doorway to enter later. SANS Institute lists this as number 3 on the Top Ten list. CVE-1999-0018 and CVE-1999-0019 report this attack. The syslog entry for sm_mon suggests this attack is really CVE-1999-0493.

5. Attack Mechanism:

This attack mechanism works by querying the portmapper for the port number used by rpc.statd, a process used to monitor systems mostly for use with NFS. Once the port number has been found, the attacker attempts a remote buffer overflow against the daemon.

Since UDP is used and the return traffic not is needed for the exploit to work, the source address could have been easily spoofed. In order for that to work the port number used by rpc.statd would have to be known. It is possible the attacker has spoofed the source address; however the first call to UDP port 111 suggests the program used to launch the attack wants to know the port number before attempting the overflow.

If the remote overflow was successful, most likely a command was executed on our NFS server.

6. Correlations:

This particular detect is not new. Buffer overflows against rpc.statd are well known and are considered in the top ten list (www.sans.org/topten.htm). The CVE numbers listed above are reports previously issued on the subject.
7. Evidence of active targeting:

This looks like active targeting. The only traffic we have coming in for SCANNER.OTHER.NET at this time is against our NFS server and is a remote exploit against a daemon used with NFS.

8. Severity:

Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures)

Criticality: 5 (The destination host is a core NFS server)
Lethality: 5 (Root access over the net)
System Countermeasures: 4 (Modern OS, all patches, additional security)
Network Countermeasures: 1 (Little to no protection from firewalls)

Severity = (5 + 5) - (4 + 1) = 5.

NOTE: We can not really tell if this attempt was successful from the network scan. No traffic suggesting an active session following the attack doesn't mean the server is in a secure state.

9. Defensive recommendation:

Recommendation is to implement a packet filter and firewall to deny all packets requesting rpc and nfs services from entering and leaving our network. Additionally, we should do a full security scan of our NFS server looking for evidence of a compromise. Finally, we should review our need for NFS, our export filesystems' characteristics, consider using secure rpc, and verify patching and logging procedures are being followed. Finally, reset all passwords on NFS server, with proactive composition checking.

10. Multiple choice test question:

22:32:27.256028 SCANNER.OTHER.NET.783 > NFS_SERVER.MY.NET.sunrpc: udp 56 (ttl 64, id 41021)
22:32:27.257397 NFS_SERVER.MY.NET.sunrpc > SCANNER.OTHER.NET.783: udp 28 (ttl 64, id 49957)
22:32:27.262975 SCANNER.OTHER.NET.862 > NFS_SERVER.MY.NET.1011: udp 1112 (ttl 64, id 64250)
22:32:27.274461 NFS_SERVER.MY.NET.1011 > SCANNER.OTHER.NET.862: udp 32 (ttl 64, id 49958)

Given this tcpdump, which is not likely:

a) SCANNER.OTHER.NET attempted a remote buffer overflow attack against DNS_SERVER.
b) A UDP datagram of size 1112 is normal.
c) SCANNER.OTHER.NET is querying NFS_SERVER.MY.NET for rpcinfo
d) SCANNER.OTHER.NET and NFS_SERVER.MY.NET are physically close to each other.

Answer: b

Detect 3 - A rpcinfo scan

The following is Snort output data:

[**] RPC Info Query [**]
08/12-22:49:58.851419 SCANNER.OTHER.NET:1008 -> WORKSTATION-01.MY.NET:111
TCP TTL:64 TOS:0x0 ID:54498 DF
*****PA* Seq: 0x86873519 Ack: 0x45020017 Win: 0x00000000 ... (until WORKSTATION-N.MY.NET)

The following is tcpdump output data:

22:49:58.695935 SCANNER.OTHER.NET.1008 > WORKSTATION-01.MY.NET.sunrpc: S 2257007896:2257007896(0) win 16384 (DF) (ttl 64, id 54495)
22:49:58.848949 SCANNER.OTHER.NET.1008 > WORKSTATION-01.MY.NET.sunrpc: . ack 1 win 17520 (DF) (ttl 64, id 54497)
22:49:58.851392 SCANNER.OTHER.NET.1008 > WORKSTATION-01.MY.NET.sunrpc: P 1:45(44) ack 1 win 17520 (DF) (ttl 64, id 54497)
22:49:59.019823 WORKSTATION-01.MY.NET.sunrpc > SCANNER.OTHER.NET.1008: . ack 45 win 24820 (DF) (ttl 64, id 54498)
22:49:59.204561 WORKSTATION-01.MY.NET.sunrpc > SCANNER.OTHER.NET.1008: P 1:1093(1092) ack 45 win 24820 (DF) (ttl 64, id 54498)
22:49:59.400318 SCANNER.OTHER.NET.1008 > WORKSTATION-01.MY.NET.sunrpc: . ack 1093 win 17520 (DF) (ttl 64, id 54501)
22:50:00.347377 SCANNER.OTHER.NET.1008 > WORKSTATION-01.MY.NET.sunrpc: P 45:45(0) ack 1093 win 17520 (DF) (ttl 64, id 54500)
22:50:00.495876 WORKSTATION-01.MY.NET.sunrpc > SCANNER.OTHER.NET.1008: . ack 46 win 24820 (DF) (ttl 64, id 54506)
22:50:00.502596 WORKSTATION-01.MY.NET.sunrpc > SCANNER.OTHER.NET.1008: P 1093:1093(0) ack 46 win 24820 (DF) (ttl 64, id 54515)
22:50:00.502790 SCANNER.OTHER.NET.1008 > WORKSTATION-01.MY.NET.sunrpc: . ack 1094 win 17520 (DF) (ttl 64, id 54508) ...

1. Source of trace:

A network designed specifically for analyzing intrusion attempts with little or no network countermeasures.

2. Detect was generated by:

Detected by Snort (The Lightweight Network Intrusion Detection System) with a full ruleset, and tcpdump.

3. Probability the source address was spoofed:

The probability is low, because the attacker wants to see the response and TCP is used. This is a scan against our entire network.
4. Description of the attack:

The attacker is scanning all our hosts to determine which rpc services they are offering. This appears to be a reconnaissance, and could be followed up by targeted attacks against vulnerable systems. Possible follow ups are CVE-1999-0003, 0008, 0208, 0212, 0228, 0320, 0353, 0493, 0687, 0696, 0900, 0969, and/or 0974; Additionally CAN-1999-0078, 0195, 0568, 0613, 0625, 0632, 0795, and/or CAN-2000-0114, 0508, and/or 0544.

5. Attack Mechanism:

This attack mechanism works by requesting a dump() from a host's portmapper. This provides a listing of the rpc programs with their versions, protocols, ports, and names listed. The goal here is to patrol for vulnerable rpc services and launch a targeted attack in the near future.

6. Correlations:

This particular detect is not new. Using system commands such as % rpcinfo -p (hostname) give out this information. Many rpc services are vulnerable to remote buffer overflow attacks. The CVE numbers listed above are reports previously issued on the subject.

7. Evidence of active targeting:

The attacker is just starting active targeting by getting a listing of the rpc services available on our hosts. Once the attacker has analyzed this information, we could see highly targeted attempts against our hosts.

8. Severity:

Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures)

Criticality: 5 (The scan is across our entire network)
Lethality: 2 (This attack is acquiring information about our network)
System Countermeasures: 1 (At least one system has little or not protection)
Network Countermeasures: 1 (Little to no protection from firewalls)

Severity = (5 + 2) - (1 + 1) = 5.

9. Defensive recommendation:

Recommendation is to implement a packet filter and firewall to deny all packets requesting a dump() from our portmappers. Additionally, we should scan our hosts for rpc services, eliminate all unneeded rpc services, fully patch all systems, and check key systems for evidence of compromise. Finally, we should install secure rpc on our systems, and verify patching and logging procedures are being followed.

10. Multiple choice test question:

This tcpdump trace shows:

- A call to WORKSTATION-01.MY.NET's portmapper for dump().
- A call to WORKSTATION-01.MY.NET's portmapper for getport().
- WORKSTATION-01 and SCANNER are sync-ing rpc maps
- A call to WORKSTATION-01.MY.NET's portmapper for dump().

Answer: a

Detect 4 - NMAP Scan

The following is Snort output data:

```
[**] IDS162 - PING Nmap2.36BETA [**]
08/12-22:59:12.196318 SCANNER.OTHERNET > WORKSTATION-01
ICMP TTL:49 TO:0x0 ID:48343
ID:57355 Seq:0 ECHO

[**] spp_portscan: PORTSCAN DETECTED from SCANNER.OTHER_NET [**]
08/12-22:59:12.36BETA [**]
08/12-22:59:12.593348 SCANNER.OTHERNET > WORKSTATION-01
TCP TTL:48 TO:0x0 ID:50569
**S***** Seq: 0x8BC78C11 Ack: 0x0 Win: 0x400

[**] IDS58 - BACKDOOR ATTEMPT- PossibleSilencer-Hebeex-Doly [**]
08/12-22:59:12.593348 SCANNER.OTHERNET > WORKSTATION-01
TCP TTL:48 TO:0x0 ID:51795
**S***** Seq: 0x8BC78C11 Ack: 0x0 Win: 0x400
```
**IDS80** - **BACKDOOR ATTEMPT-Netbus/GabanBus [**]
TCP TTL:48 TOS:0 ID:40164
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**AOL Chat Data Logged [**]
TCP TTL:64 TOS:0 ID:5072
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**AOL Chat Data Logged [**]
TCP TTL:16 TOS:0 ID:8474
**R** Seq: 0x0 Ack: 0xBC7C8C12 Win: 0x0

**BACKDOOR ATTEMPT-SocketsDeTroie [**]
TCP TTL:64 TOS:0 ID:23095
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS80** - **BACKDOOR ATTEMPT-Netbus/GabanBus [**]
TCP TTL:48 TOS:0 ID:20371
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS80** - **IDS69** - **AOL Chat Data Logged [**]
08/12-22:59:12.859777 SCANNER.OTHER.NET:43645 -> WORKSTATION-01:1032
TCP TTL:48 TOS:0 ID:18767
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS80** - **IDS69** - **AOL Chat Data Logged [**]
08/12-22:59:12.859777 SCANNER.OTHER.NET:43645 -> WORKSTATION-01:1032
TCP TTL:48 TOS:0 ID:58887
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS84** - **BACKDOOR ATTEMPT-Masters Paradise [**]
TCP TTL:48 TOS:0 ID:46593
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS84** - **BACKDOOR ATTEMPT-Masters Paradise [**]
TCP TTL:48 TOS:0 ID:23095
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS84** - **IDS69** - **AOL Chat Data Logged [**]
TCP TTL:48 TOS:0 ID:54986
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS84** - **IDS69** - **AOL Chat Data Logged [**]
TCP TTL:48 TOS:0 ID:58887
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS84** - **IDS69** - **AOL Chat Data Logged [**]
TCP TTL:48 TOS:0 ID:18259
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS52** - **BACKDOOR ATTEMPT-Psyber Streaming Server [**]
TCP TTL:48 TOS:0 ID:20326
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS63** - **BACKDOOR ATTEMPT-Schoolbus 1.0 [**]
TCP TTL:48 TOS:0 ID:21112
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS57** - **BACKDOOR ATTEMPT-Socket 23 [**]
TCP TTL:48 TOS:0 ID:52775
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS57** - **BACKDOOR ATTEMPT-Socket 23 [**]
TCP TTL:48 TOS:0 ID:52775
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS89** - **BACKDOOR ATTEMPT-Backorifice [**]
TCP TTL:48 TOS:0 ID:3200
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS60** - **BACKDOOR ATTEMPT-Der Spaeher 3 [**]
TCP TTL:48 TOS:0 ID:16086
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS60** - **BACKDOOR ATTEMPT-Hidden Port 2.0 [**]
TCP TTL:48 TOS:0 ID:5896
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS60** - **BACKDOOR ATTEMPT-Yahoo! Messenger Exploit Attempt [**]
TCP TTL:48 TOS:0 ID:34627
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS60** - **BACKDOOR ATTEMPT-Shivka-Burka [**]
TCP TTL:48 TOS:0 ID:16086
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

**IDS60** - **BACKDOOR ATTEMPT-Hidden Port 2.0 [**]
TCP TTL:48 TOS:0 ID:5896
**S***** Seq: 0xBC7C8C11 Ack: 0x0 Win: 0x400

© SANS Institute 2000 - 2005
Author retains full rights.
The following is tcpdump output data:

TCP TTL:48 TOs:0x0 ID:11675
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] IDS36 - BACKDOOR ATTEMPT-WinCrash [**]
TCP TTL:48 TOs:0x0 ID:58487
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] BACKDOOR ATTEMPT-TCPShell - NIX Backdoor Attempt [**]
TCP TTL:48 TOs:0x0 ID:49475
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] BACKDOOR ATTEMPT-BO Jammer Killah V [**]
TCP TTL:48 TOs:0x0 ID:38323
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] BACKDOOR ATTEMPT-Doly Trojan 1.6 [**]
TCP TTL:48 TOs:0x0 ID:24453
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] OVERFLOW - Possible attempt at MS Print Services [**]
TCP TTL:48 TOs:0x0 ID:48831
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] BACKDOOR ATTEMPT-Doly Trojan 1.5 [**]
TCP TTL:48 TOs:0x0 ID:55625
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] MIS-C-Attempted Sun RPC high port access [**]
TCP TTL:48 TOs:0x0 ID:62966
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] IDS45 - BACKDOOR ATTEMPT-The Thing [**]
TCP TTL:48 TOs:0x0 ID:33224
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] IDS100 - BACKDOOR ATTEMPT- FTP99CMF [**]
TCP TTL:48 TOs:0x0 ID:25698
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] MISC-WinGate-8080-Attempt [**]
08/12-22:59:15.128775 SCANNER.OTHER.NET:43645 -> WORKSTATION-01:8080
TCP TTL:48 TOs:0x0 ID:58886
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] IDS94 - BACKDOOR ATTEMPT- HackersParadise [**]
08/12-22:59:15.133957 SCANNER.OTHER.NET:43645 -> WORKSTATION-01:456
TCP TTL:48 TOs:0x0 ID:318
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] BACKDOOR ATTEMPT- Attack FTP / Satans Backdoor [**]
08/12-22:59:15.144682 SCANNER.OTHER.NET:43645 -> WORKSTATION-01:6666
TCP TTL:48 TOs:0x0 ID:16341
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] IDS52 - BACKDOOR ATTEMPT- Vresar Streaming Server [**]
08/12-22:59:15.190345 SCANNER.OTHER.NET:43645 -> WORKSTATION-01:1509
TCP TTL:48 TOs:0x0 ID:39215
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] IDS41 - BACKDOOR ATTEMPT-Transcout [**]
TCP TTL:48 TOs:0x0 ID:30101
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] IDS34 - BACKDOOR ATTEMPT-XTCP2 [**]
08/12-22:59:15.446147 SCANNER.OTHER.NET:43645 -> WORKSTATION-01:5550
TCP TTL:48 TOs:0x0 ID:37601
**S***** Seq: 0x8BC7C8C11 Ack: 0x0 Win: 0x400

[**] IDS05 - SCAN-Possible NMAP Fingerprint attempt [**]
08/12-22:59:15.484652 SCANNER.OTHER.NET:43645 -> WORKSTATION-01:21
TCP TTL:48 TOs:0x0 ID:64258
**S***** Seq: 0x8FD7E7A7 Ack: 0x0 Win: 0x400
TCP Options => WS: 10 NOP MSS: 265 TS: 106119567 0 EOL EOL

[**] IDS28 - PING NMAP TCP [**]
08/12-22:59:15.485004 SCANNER.OTHER.NET:43645 -> WORKSTATION-01:21
TCP TTL:48 TOs:0x0 ID:51733
**S***** Seq: 0x8FD7E7A7 Ack: 0x0 Win: 0x400
TCP Options => WS: 10 NOP MSS: 265 TS: 106119567 0 EOL EOL

[**] IDS28 - PING NMAP TCP [**]
08/12-22:59:15.486044 SCANNER.OTHER.NET:43645 -> WORKSTATION-01:11
TCP TTL:48 TOs:0x0 ID:61711
**S***** Seq: 0x8FD7E7A7 Ack: 0x0 Win: 0x400
TCP Options => WS: 10 NOP MSS: 265 TS: 106119567 0 EOL EOL

The following is tcpdump output data:
A network designed specifically for analyzing intrusion attempts with little or no network countermeasures.

This information can then be used to launch a targeted attack.

Little or no network countermeasures.

This a reconnaissance, and most likely will be followed up by any number of targeted attempts.

Note: in some case the sequence numbers do not change.

The attacker is using tcpdump.

TCP is used.

This is a reconnaissance, and most likely will be followed up by any number of targeted attempts.

Note: in some case the source port numbers and the sequence numbers do not change.

The attacker is interested in learning which services we have running and what operating system we are using.

This is an attempt to hide the scan since it only allows the first two steps of the three-way tcp handshake to take place.

The attacker is interested in learning which services we have running and what operating system we are using. This information can then be used to launch a targeted attack.

A complete FAQ and manual for nmap can be found on the website.

A complete FAQ and manual for nmap can be found on the website.
This particular detect is not new. Host scanning is well known and several new tools are written each day. Nmap is considered a best of breed among various communities. Nmap also supports several stealth scanning modes.

7. Evidence of active targeting:

The attacker is just starting active targeting by mapping our host. We could see a targeted attack in the near future pertaining to our hosts' operating systems and running daemon processes.

8. Severity:

Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures)

Criticality: 2 (UNIX Desktop)
Lethality: 2 (This attack is acquiring information about our network)
System Countermeasures: 3 (Modern OS, might be missing patches)
Network Countermeasures: 1 (Little to no protection from firewalls)

Severity = (2 + 2) - (3 + 1) = 0.

9. Defensive recommendation:

Recommendation is to implement a packet filter and firewall to deny all packets connection requests originating from outside our network, and block all known trojan ports. Filters should be put in place to block malformed packets (XMAS, FIN/SYN scans, etc) and IDS signatures put in place. Additionally, we should double check which daemons are running on our workstations, turn off unneeded ones, and replace needed ones with more secure replacements if possible (i.e. use secure-shell instead of ftp and telnet, use TCP Wrappers, etc). Finally, we may want to verify patching and logging procedures are being followed.

10. Multiple choice test question:

This tcpdump trace shows a scan. Which services are WORKSTATION-01 running:

a) ftp, ssh
b) ssh, telnet
c) ftp, telnet
d) rpc, ssh

Answer: c

Detect 5 - nestea attack

The following is Snort output data:

[**] Tiny Fragments - Possible Hostile Activity[**]
08/12-23:22:05.956309 10.1.1.1 -> 10.1.1.1
UDP TTL:64 TID:0x0 1D:242 MF
Frag Offset: 0x0  Frag Size: 0x12
... 1499 more like this follow with timestamps slowly increasing..

The following is tcpdump output data:

23:22:05.956297 TARGET.MY.NET.51328 > TARGET.MY.NET.44411: udp 10 (frag 242:1800+) (ttl 64)
23:22:05.956560 TARGET.MY.NET > TARGET.MY.NET: (frag 242:116848) (ttl 64)
23:22:05.957042 |udp| (frag 242:26400+) (ttl 64, optlen=40 EOL-39)
23:22:05.969750 TARGET.MY.NET.51328 > TARGET.MY.NET.44411: udp 10 (frag 242:1800+) (ttl 64)
23:22:05.970099 TARGET.MY.NET > TARGET.MY.NET: (frag 242:116848) (ttl 64)
23:22:05.970582 |udp| (frag 242:26400+) (ttl 64, optlen=40 EOL-39)
23:22:05.989554 TARGET.MY.NET.51328 > TARGET.MY.NET.44411: udp 10 (frag 242:1800+) (ttl 64)
23:22:05.989760 TARGET.MY.NET > TARGET.MY.NET: (frag 242:116848) (ttl 64)
23:22:05.990046 |udp| (frag 242:26400+) (ttl 64, optlen=40 EOL-39)
23:22:06.009658 TARGET.MY.NET.51328 > TARGET.MY.NET.44411: udp 10 (frag 242:1800+) (ttl 64)
23:22:06.009997 TARGET.MY.NET > TARGET.MY.NET: (frag 242:116848) (ttl 64)
23:22:06.010275 |udp| (frag 242:26400+) (ttl 64, optlen=40 EOL-39)
... many more like this follow with timestamps slowly increasing..

1. Source of trace:

A network designed specifically for analyzing intrusion attempts with little or no network countermeasures.

2. Detect was generated by:
3. Probability the source address was spoofed:

The probability is 100% that the source address is spoofed. We should never see UDP datagrams with the same source and destination addresses in our network. Furthermore, there are some serious issues with fragmentation going on here.

4. Description of the attack:

The attacker is sending spoofed UDP datagrams with negative offset fragmentation to a host on our network. The intent here is most likely a Denial of Service attack against unpatched IP stacks which can not handle negative fragmentation offsets. Since the fragment id is 242 this is most likely a teardrop DOS attack or a variant like nestea, nestea2 or land.

These attacks are reported in CAN-1999-0015, 0104, 0257, and 0258.

5. Attack Mechanism:

This attack mechanism works by sending datagrams with fragments which overlap each other using negative offsets. Many older IP stacks could not handle these pathological offsets and would crash or hang. Additionally, this attack attempts to flood the host by sending the final segment several hundred times. The idea is fill up the buffers on the target host and/or cause it to reference a invalid memory address.

6. Correlations:

This particular detect is not new. Many DOS using tricks with fragmentation exist and are well known. The CVE numbers listed above are reports previously issued on the subject, in addition, BUGTRQAQ:19981023, BUGTRAQ19990909, and CERT:CA-97.28.Tear1dop_Land are sources of information. Also, the fragment id of 242 is an intrusion detection give me.

7. Evidence of active targeting:

Since this system is a fully-patched modern OS with additional security, no evidence of active targeting exists. Additionally, this datagram did not travel across any Cisco equipment.

8. Severity:

Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures)

Criticality: 2 (Fully-patched UNIX desktop with modern OS)
Lethality: 1 (This attack is highly unlikely to succeed)
System Countermeasures: 4 (Modern OS, all patches, additional security)
Network Countermeasures: 1 (Little to no protection from firewalls)

Severity = (2 + 1) - (4 + 1) = 0.

NOTE: Against a UNIX box today this has little threat.

9. Defensive recommendation:

Recommended is to implement a packet filter and firewall to deny all fragmented UDP datagrams, small fragments, and packets with source addresses matching internal hosts from into our network. Additionally, we should implement anti-spoofing filters. Finally, we should plan to rebuild our firewalls with a first 'deny all' stance.

10. Multiple choice test question:

This tcpdump trace shows evidence of:

a) Packets entering your network from daemon internet.
b) Spoofed source address.
c) Port scanning.
d) OS Detection.

Answer: b

Evaluate an Attack - Xsh0k.c

Location of attack:
Description of attack:

On 2/5/00 nتوفر (NeURaL Col.LaPaC CEw) released a Denial of Service Attack for Xwindow to rootshell.

Xsh0k.c works by flooding the Xwindow port (6000) on the target host with TCP connection requests. The idea is grab all available sockets from the Xserver (see accept and listen socket calls) in hopes of denying the local user from starting new programs (clients), slow down their session, and/or even crash it.

Three issues exist with the code as it is written. First, when the program runs, it will eventually run out of file descriptors and exit (line 139). So, this attack when run by itself (unmodified) will not work. Second, this attack would need to be run in a distributed fashion in order to hose the target host (open file limits). Thirdly, this DOS uses TCP and requires the three-way handshake to complete. In other words, an attacker can not hide when using this attack.

A defense against this attack is simple: Deny packets with a destination port of 6000 from entering your network. Better yet, log them and run them to ground. You are likely to find a compromised Linux box on the Internet whose owner would really like to know about it.

Annotated network trace:

Here we see the first step in the three-way handshake (SYN):

Here we see the second step in the three-way handshake (SYN/ACK):

Here we see the third step in the three-way handshake (ACK):

Now we see a bunch more:
At this point, we see a few unusual things:

About 122 connections are open.

The attacker is most likely using a for loop using a call like connect() sequentially with only one copy running. Additionally, the timestamps are incrementing evenly.

No communication is taking place between the open connections yet.

Next we see the open connections starting to close in a usual manner (active close):

About 122 connections are open.

The attacker is most likely using a for loop using a call like connect() sequentially with only one copy running. Additionally, the timestamps are incrementing evenly.

No communication is taking place between the open connections yet.
Based on the time and the number of connections, we can speculate on more resources (file descriptors) than our attacker.

At this point (really the end), we can draw some conclusions:

- Based on the time difference 11:26:44.78108 - 11:26:45.275392, it didn't take long for the connections to tear down.
- Based on the time and the number of connections, we can speculate on SCANNER's host processor and network speeds.
- Something went wrong, this is very usual behavior. Perhaps the program on SCANNER died - i.e. ran out of open sockets. If so, our Xserver has more resources (file descriptors) than our attacker.
A generalized signature for this program is a lot of connection requests to port 6000 without any data being exchanged and eventually the connections being torn down in a short period of time. Similar to a SYN flood, only not anonymous. IMHO, nothing to worry about yet...

Moral of the Story: If you are trying to use up someone else's resources, make sure you have enough of your own first.

"Analyze This" Scenario

Background

My hypothetical organization, ACME Security Services, was asked to provide a bid for security services for a facility. ACME asked me to run a Snort system with a fairly standard rulebase for a month. The system suffered from random power failures and full disks, so the collected data represents a statistical sampling. For my analysis, ACME asked me to concentrate on system compromises and network outages.

Assumptions

Since this is a scenario based question, I'm going to draw some assumptions:

- ACME placed the Snort system somewhere in the facility where it will scar interesting traces.
- ACME is interested in scooping for the amount of work involved to secure this facility and has the expertise to do so.
- ACME needs to motivate the facility on the need for security by giving at least one example of a compromise or great potential for one.
- ACME wants a brief high-level overview of my analysis and wants traces as exhibits. ACME expects to call upon me for detailed information if the needs exists.

Analysis Report

ACME Security Services has a large task to face if they win the bid. This facility has a Class B Network (65535 possible hosts - subnetting costs) and is experiencing a significant amount of reconnaissance. DNS appears to be the most popular choice. The other targets are rcp, FTP, telnet, sendmail, POP2, POP3, IMAP, Linuxconf, web servers, and web proxies. Scanning is taking place around the clock. Additionally, denial of service and public snmp traffic has been captured.

Compromises seem to have been found in both DNS and RPC. Both being accessed remotely from both internal and external hosts. This is a SANS Institute Top 10 scenario. Furthermore, neither network or system countermeasures seem to exist.

Recommendations:

- Follow SANS Security Roadmap.
- Audit MY.NET.253.12 for compromise.
- Audit MY.NET.2.203, MY.NET.253.24, MY.NET.130.94, MY.NET.97.106, MY.NET.70.127, MY.NET.143.87, MY.NET.179.78, and MY.NET.99.51 for compromise.
- Implement chroot() DNS servers, secure rcp, and TCP Wrappers.
- Inventory network and perform cost-benefit analysis based on Bruce Schneier's Attack Tree Model (www.counterpane.com/attacktrees-ddj-ft.html)
- Segment network with both internal and external firewalls, packet filtering devices, intrusion detection systems, and deploy host level security.
- Perform penetration testing.

Details to follow as exhibits (many are sniplets).

Exhibits

Troyjan Scan:

Jun 4 00:06:17 203.94.224.241:3278 -> MY.NET.60.11:27374 SYN **S*****
Jun 4 00:06:17 203.94.224.241:3280 -> MY.NET.60.11:6671 SYN **S*****
Jun 4 00:06:13 203.94.224.241:3283 -> MY.NET.60.11:456 SYN **S*****
Jun 4 00:06:18 203.94.224.241:3279 -> MY.NET.60.11:6670 SYN **S*****
Jun 4 00:06:18 203.94.224.241:3281 -> MY.NET.60.11:21554 SYN **S*****
Jun 4 00:06:16 203.94.224.241:3277 -> MY.NET.60.11:1243 SYN **S*****
Jun 4 00:06:18 203.94.224.241:3276 -> MY.NET.60.11:31377 SYN **S*****
Jun 4 00:06:18 203.94.224.241:3274 -> MY.NET.60.11:1080 SYN **S*****
Jun 4 00:06:16 203.94.224.241:3275 -> MY.NET.60.11:20034 SYN **S*****
Jun 4 00:06:16 203.94.224.241:3273 -> MY.NET.60.11:12345 SYN **S*****
Jun 4 00:06:18 203.94.224.241:3282 -> MY.NET.60.11:9400 SYN **S*****
Jun 4 00:16:51 203.94.224.241:4938 -> MY.NET.60.11:1243 SYN **S*****
Jun 4 00:16:52 203.94.224.241:4941 -> MY.NET.60.11:6671 SYN **S*****
Jun 4 00:16:51 203.94.224.241:4936 -> MY.NET.60.11:20034 SYN **S*****
Jun 4 00:16:51 203.94.224.241:4939 -> MY.NET.60.11:27374 SYN **S*****
Jun 4 00:16:51 203.94.224.241:4937 -> MY.NET.60.11:31377 SYN **S*****
Jun 4 00:16:51 203.94.224.241:4940 -> MY.NET.60.11:6670 SYN **S*****
Jun 4 00:16:51 203.94.224.241:4943 -> MY.NET.60.11:9400 SYN **S*****
Jun 4 00:16:51 203.94.224.241:4942 -> MY.NET.60.11:21554 SYN **S*****
Jun 4 00:16:52 203.94.224.241:4944 -> MY.NET.60.11:1456 SYN **S*****
Jun 4 00:16:56 203.94.224.241:4934 -> MY.NET.60.11:12345 SYN **S*****
Jun 4 00:16:56 203.94.224.241:4935 -> MY.NET.60.11:1080 SYN **S*****
Jun 4 00:16:54 203.94.224.241:4938 -> MY.NET.60.11:1243 SYN **S*****
Jun 4 00:16:54 203.94.224.241:4940 -> MY.NET.60.11:6670 SYN **S*****
Jun 4 00:16:54 203.94.224.241:4942 -> MY.NET.60.11:21554 SYN **S*****
Jun 4 00:16:54 203.94.224.241:4939 -> MY.NET.60.11:27374 SYN **S*****
Jun 4 00:16:54 203.94.224.241:4937 -> MY.NET.60.11:31377 SYN **S*****
Jun 4 00:16:54 203.94.224.241:4943 -> MY.NET.60.11:9400 SYN **S*****

POP2 Scan:

Dns Scan (with queries?):

Dns Scan w/ high source port number:

Port Scan:

Ftp Scan:

Linuxconf Scan:

Ftp Bounce Port Scan:
May 27 23:44:43 MY.NET.253.12:43746 \rightarrow MY.NET.14.1:1229 SYN **S*****
May 27 23:44:45 MY.NET.253.12:22108 \rightarrow MY.NET.14.1:113 SYN **S*****
May 27 23:44:45 MY.NET.253.12:415815 \rightarrow MY.NET.14.1:119 SYN **S*****
May 27 23:44:45 MY.NET.253.12:12158 \rightarrow MY.NET.14.1:213 SYN **S*****
May 27 23:44:45 MY.NET.253.12:120245 \rightarrow MY.NET.14.1:179 SYN **S*****
May 27 23:44:45 MY.NET.253.12:48810 \rightarrow MY.NET.14.1:2001 SYN **S*****
May 27 23:44:47 MY.NET.253.12:47851 \rightarrow MY.NET.14.1:6001 SYN **S*****
May 27 23:44:47 MY.NET.253.12:43755 \rightarrow MY.NET.14.1:7 NMAPID **SP**
May 27 23:44:47 MY.NET.253.12:43757 \rightarrow MY.NET.14.1:1 SYN **S*****
May 27 23:44:47 MY.NET.253.12:43746 \rightarrow MY.NET.14.1:1 UDP
May 27 23:44:47 MY.NET.253.12:43752 \rightarrow MY.NET.14.1:7 SYN **S*****

Scan for Sub 7:
May 26 21:21:07 24.2.169.101:1046 \rightarrow MY.NET.120.158:27374 SYN **S*****
May 26 21:21:07 24.2.169.101:1047 \rightarrow MY.NET.120.159:27374 SYN **S*****
May 26 21:21:07 24.2.169.101:1048 \rightarrow MY.NET.120.160:27374 SYN **S*****
May 26 21:21:07 24.2.169.101:1049 \rightarrow MY.NET.120.161:27374 SYN **S*****
May 26 21:21:07 24.2.169.101:1050 \rightarrow MY.NET.120.162:27374 SYN **S*****
May 26 21:21:07 24.2.169.101:1051 \rightarrow MY.NET.120.163:27374 SYN **S*****
May 26 21:21:07 24.2.169.101:1052 \rightarrow MY.NET.120.164:27374 SYN **S*****
May 26 21:21:07 24.2.169.101:1056 \rightarrow MY.NET.120.168:27374 SYN **S*****
May 26 21:21:10 24.2.169.101:1058 \rightarrow MY.NET.120.170:27374 SYN **S*****
May 26 21:21:10 24.2.169.101:1067 \rightarrow MY.NET.120.171:27374 SYN **S*****

Sendmail Scan:
May 26 18:59:22 216.32.241.15:1137 \rightarrow MY.NET.253.41:25 SYN **S*****
May 26 18:59:23 216.32.241.15:1144 \rightarrow MY.NET.6.35:25 SYN **S*****
May 26 18:59:23 216.32.241.15:1143 \rightarrow MY.NET.6.34:25 SYN **S*****
May 26 18:59:23 216.32.241.15:1141 \rightarrow MY.NET.6.32:25 SYN **S*****
May 26 18:59:23 216.32.241.15:1148 \rightarrow MY.NET.6.47:25 SYN **S*****
May 26 18:59:23 216.32.241.15:1147 \rightarrow MY.NET.253.43:25 SYN **S*****

Unknown; However a source port of 0 is a give me:
Jun 10 00:13:10 142.137.141.178:0 \rightarrow MY.NET.217.62:2014 INVALIDACK **S*R*A*

Scan for Netbus or Ultra's Telnet Trojan:
Jun 11 17:53:48 212.153.128.116:09742 \rightarrow MY.NET.209.41:22345 SYN **S*****
Jun 11 17:53:46 212.153.128.116:4696 \rightarrow MY.NET.208.250:12345 SYN **S*****
Jun 11 17:53:46 212.153.128.116:4697 \rightarrow MY.NET.208.251:12245 SYN **S*****
Jun 11 17:53:46 212.153.128.116:11798 \rightarrow MY.NET.209.98:12345 SYN **S*****

© SANS Institute 2000 - 2005 Author retains full rights.
### Pop3 Scan:

<table>
<thead>
<tr>
<th>Date</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Protocol</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 15 05:31:29</td>
<td>207.189.129.1:2666</td>
<td>-&gt; MY.NET.1:211:110</td>
<td>SYN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jun 15 05:31:29</td>
<td>207.189.129.1:2666</td>
<td>-&gt; MY.NET.1:212:110</td>
<td>SYN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Trojan 44767 or a scan for alive hosts (expecting ICMP port unreachable):

<table>
<thead>
<tr>
<th>Date</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Protocol</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 16 13:02:18</td>
<td>212.29.82.197:4584</td>
<td>-&gt; MY.NET.97:1244767</td>
<td>UDP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jun 16 13:02:18</td>
<td>212.29.82.197:4585</td>
<td>-&gt; MY.NET.97:1344767</td>
<td>UDP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jun 16 13:02:18</td>
<td>212.29.82.197:4586</td>
<td>-&gt; MY.NET.97:1444767</td>
<td>UDP</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Telnet Scan:

<table>
<thead>
<tr>
<th>Date</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Protocol</th>
<th>State</th>
</tr>
</thead>
</table>

### Scan for Web Server on 8080, an user installed proxy server, or WinGate 8080:

<table>
<thead>
<tr>
<th>Date</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Protocol</th>
<th>State</th>
</tr>
</thead>
</table>

### Scan for well known services (web, telnet, sendmail, and trojans):

<table>
<thead>
<tr>
<th>Date</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Protocol</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 12 05:51:09</td>
<td>148.204.183.85:17673</td>
<td>-&gt; MY.NET.60:14:6000</td>
<td>SYN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jun 12 05:51:09</td>
<td>148.204.183.85:17675</td>
<td>-&gt; MY.NET.60:14:53</td>
<td>SYN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jun 12 05:51:09</td>
<td>148.204.183.85:17731</td>
<td>-&gt; MY.NET.60:14:31337</td>
<td>SYN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Trojan 327727 or a scan for alive hosts (expecting ICMP port unreachable):

<table>
<thead>
<tr>
<th>Date</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Protocol</th>
<th>State</th>
</tr>
</thead>
</table>

### Trojan 6972,4,6 (or host ping):

<table>
<thead>
<tr>
<th>Date</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Protocol</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 13 13:48:08</td>
<td>207.25.79.227:19510</td>
<td>-&gt; MY.NET.156:118:6974</td>
<td>UDP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jun 13 13:48:08</td>
<td>207.25.79.227:11630</td>
<td>-&gt; MY.NET.156:121:6974</td>
<td>UDP</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

© SANS Institute 2000 - 2005

Author retains full rights.
Bean for ftp and imap:
Jun 22 20:58:09 212.25.68.195:1658 -> MY.NET.106.158:23 SYN **S*****
Jun 22 20:58:11 212.25.68.195:2320 -> MY.NET.140.83:23 SYM **S*****
Jun 22 20:58:09 212.25.68.195:2321 -> MY.NET.140.83:23 SYM **S*****
Jun 22 20:58:09 212.25.68.195:1677 -> MY.NET.105.232:23 SYM **S*****
Jun 22 20:58:11 212.25.68.195:2336 -> MY.NET.105.232:143 SYM **S*****
Jun 22 20:58:09 212.25.68.195:1676 -> MY.NET.105.136:23 SYM **S*****
Jun 22 20:58:11 212.25.68.195:2335 -> MY.NET.105.136:143 SYM **S*****

Scan for ftp and pop3:
Jun 22 20:58:18 212.25.68.195:2576 -> MY.NET.151.66:110 SYM **S*****
Jun 22 20:58:19 212.25.68.195:2571 -> MY.NET.151.174:110 SYM **S*****
Jun 22 20:58:19 212.25.68.195:2577 -> MY.NET.151.174:110 SYM **S*****
Jun 22 20:58:19 212.25.68.195:2578 -> MY.NET.151.174:110 SYM **S*****
Jun 22 20:58:19 212.25.68.195:2579 -> MY.NET.151.174:110 SYM **S*****
Jun 22 20:58:18 212.25.68.195:2580 -> MY.NET.141.250:110 SYM **S*****

Scan for imap and pop3:
Jun 22 20:58:29 212.25.68.195:2300 -> MY.NET.143.106:143 SYM **S*****
Jun 22 20:58:27 212.25.68.195:2571 -> MY.NET.151.174:110 SYM **S*****
Jun 22 20:58:29 212.25.68.195:2299 -> MY.NET.151.174:143 SYM **S*****
Jun 22 20:58:29 212.25.68.195:2572 -> MY.NET.10.55:110 SYM **S*****
Jun 22 20:58:29 212.25.68.195:2298 -> MY.NET.10.55:143 SYM **S*****
Jun 22 20:58:27 212.25.68.195:2573 -> MY.NET.156.73:110 SYM **S*****

Scan for ftp:
Jun 22 22:36:49 213.188.8.45:2819 -> MY.NET.205.94:21 SYM **S*****
Jun 22 22:36:49 213.188.8.45:2820 -> MY.NET.206.158:21 SYM **S*****
Jun 22 22:36:49 213.188.8.45:2800 -> MY.NET.202.234:21 SYM **S*****
Jun 22 22:36:50 213.188.8.45:2829 -> MY.NET.208.106:21 SYM **S*****
Jun 22 22:36:51 213.188.8.45:2832 -> MY.NET.208.42:21 SYM **S*****
Jun 22 22:36:51 213.188.8.45:2807 -> MY.NET.203.178:21 SYM **S*****
Jun 22 22:36:51 213.188.8.45:2805 -> MY.NET.203.154:21 SYM **S*****
Jun 22 22:36:53 213.188.8.45:2842 -> MY.NET.201.78:21 SYM **S*****
Jun 22 22:36:54 213.188.8.45:2833 -> MY.NET.208.74:21 SYM **S*****

Interesting Snort Alert Messages:
05/22-08:38:57.992872 [**] SYN-FIN scan! [**] 142.150.225.137:53 -> MY.NET.1.27:53
05/25-19:37:58.521149 [**] Tiny Hostile Fragments - Possible Hostile Activity [**] 24.3.7.221 -> MY.NET.70.121
05/25-20:51:50.118442 [**] Watchlist 000222 NET-NCFC [**] 159.226.45.3:2013 -> MY.NET.253.4125
05/27-23:44:47.358118 [**] Probable NMAP fingerprint attempt [**] MY.NET.253.124:3758 -> MY.NET.14.11
05/27-23:44:47.358126 [**] NMAP TCP ping! [**] MY.NET.253.124:3758 -> MY.NET.14.11
05/27-23:44:47.542862 [**] Attempted Sun RPC high port access [**] 205.185.153:10000 -> MY.NET.217.2:32771
05/27-23:50:44.510484 [**] WinGate 8080 Attempt [**] 24.3.7.221 -> MY.NET.253.105:8080
05/28-14:11:49.399244 [**] SRRPC highport access! [**] MY.NET.253.124:3749 -> MY.NET.16.5:32771
06/22-20:58:01.238486 [**] External RPC call [**] 212.25.68.195:637 -> MY.NET.6.15:111

Fishing For RPC:
[**] External RPC call [**]
129.49.163.74:1005 -> MY.NET.6.15:111
129.49.163.74:882 -> MY.NET.15.127:111
129.49.163.74:802 -> MY.NET.100.130:111
129.49.163.74:802 -> MY.NET.100.130:111

Possible compromises of rpc.ttdbserv, rpc.nisd, or a showmount -e scan:
[**] SRRPC highport access! [**] 128.10.141:23 -> MY.NET.2.203:32771
192.102.49.3:25 -> MY.NET.130.94:32771
199.60.228.130:7000 -> MY.NET.97.106:32771
205.223.1.120:42455 -> MY.NET.12.53:32771
207.253.256:20 -> MY.NET.70.120:111
208.226.167.19:21 -> MY.NET.143.87:32771
24.13.123.8:1156 -> MY.NET.179.78:32771
24.13.123.8:3708 -> MY.NET.179.78:32771

Jun 12 13:48:12 207.25.79.227:30010 -> MY.NET.156.110:6974 UDP

© SANS Institute 2000 - 2005
Author retains full rights.
Key fingerprint = AF19 FA27 2F94 998D FDB5 DE3D F8B5 06E4 A169 4E46
MY.NET.253.12:43750 -> MY.NET.16.9:32771
MY.NET.253.12:43750 -> MY.NET.16.9:32771
MY.NET.253.12:43750 -> MY.NET.16.9:32771
MY.NET.253.12:43750 -> MY.NET.16.9:32771
MY.NET.253.12:43750 -> MY.NET.16.9:32771
MY.NET.253.12:43750 -> MY.NET.19.0:32771
MY.NET.253.12:43750 -> MY.NET.19.1:32771
MY.NET.253.12:43750 -> MY.NET.19.1:32771
MY.NET.253.12:43750 -> MY.NET.19.2:32771
MY.NET.253.12:43750 -> MY.NET.19.3:32771
MY.NET.253.12:43750 -> MY.NET.19.4:32771
MY.NET.253.12:43750 -> MY.NET.19.5:32771
MY.NET.253.12:43750 -> MY.NET.19.6:32771
MY.NET.253.12:43750 -> MY.NET.19.7:32771
MY.NET.253.12:43750 -> MY.NET.19.8:32771
MY.NET.253.12:43750 -> MY.NET.19.9:32771
MY.NET.253.12:43750 -> MY.NET.19.10:32771
MY.NET.253.12:43750 -> MY.NET.19.11:32771
MY.NET.253.12:43750 -> MY.NET.19.12:32771
MY.NET.253.12:43750 -> MY.NET.19.14:32771
MY.NET.253.12:43750 -> MY.NET.19.15:32771
MY.NET.253.12:43750 -> MY.NET.19.16:32771
MY.NET.253.12:43750 -> MY.NET.19.17:32771
MY.NET.253.12:43750 -> MY.NET.19.18:32771
MY.NET.253.12:43750 -> MY.NET.19.19:32771
MY.NET.253.12:43750 -> MY.NET.19.20:32771
MY.NET.253.12:43750 -> MY.NET.19.21:32771
MY.NET.253.12:43750 -> MY.NET.19.22:32771
MY.NET.253.12:43750 -> MY.NET.19.23:32771
# Upcoming Training

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Location</th>
<th>Start Date - End Date</th>
<th>Event Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Chicago 2019</td>
<td>Chicago, IL</td>
<td>Aug 19, 2019 - Aug 24, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Canberra Spring 2019</td>
<td>Canberra, Australia</td>
<td>Sep 02, 2019 - Sep 21, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Network Security 2019</td>
<td>Las Vegas, NV</td>
<td>Sep 09, 2019 - Sep 16, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Oslo September 2019</td>
<td>Oslo, Norway</td>
<td>Sep 09, 2019 - Sep 14, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Dallas Fall 2019</td>
<td>Dallas, TX</td>
<td>Sep 23, 2019 - Sep 28, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS London September 2019</td>
<td>London, United Kingdom</td>
<td>Sep 23, 2019 - Sep 28, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Santa Monica 2019</td>
<td>Santa Monica, CA</td>
<td>Oct 21, 2019 - Oct 26, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>Community SANS Toronto SEC503</td>
<td>Toronto, ON</td>
<td>Nov 04, 2019 - Nov 09, 2019</td>
<td>Community SANS</td>
</tr>
<tr>
<td>SANS Frankfurt December 2019</td>
<td>Frankfurt, Germany</td>
<td>Dec 09, 2019 - Dec 14, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Security East 2020</td>
<td>New Orleans, LA</td>
<td>Feb 01, 2020 - Feb 08, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Norfolk 2020</td>
<td>Norfolk, VA</td>
<td>Mar 16, 2020 - Mar 21, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>Online</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
<tr>
<td>SANS SelfStudy</td>
<td>Books &amp; MP3s Only</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>