Global Information Assurance Certification Paper

SANS Security DC 2000 Practical Assignment – GIAC Intrusion Detection Curriculum

Eric Brelsford
9/11/00

Table of Contents:

Assignment 1 – Network Traces

Detect #1 – RPC Scan (Portmapper)

```
Aug 14 17:03:17 195.251.213.58:111 -> a.b.c.64:111 SYNFIN **SF****
Aug 14 17:03:17 195.251.213.58:111 -> a.b.c.65:111 SYNFIN **SF****
Aug 14 17:03:17 195.251.213.58:111 -> a.b.c.66:111 SYNFIN **SF****
Aug 14 17:03:17 195.251.213.58:111 -> a.b.c.67:111 SYNFIN **SF****
Aug 14 17:03:17 195.251.213.58:111 -> a.b.c.68:111 SYNFIN **SF****
Aug 14 17:03:17 195.251.213.58:111 -> a.b.c.69:111 SYNFIN **SF****
Aug 14 17:03:17 195.251.213.58:111 -> a.b.c.70:111 SYNFIN **SF****
Aug 14 17:03:17 195.251.213.58:111 -> a.b.c.71:111 SYNFIN **SF****

[**] SCAN-SYN FIN [**]
08/14-17:03:17.596759 195.251.213.58:111 -> a.b.c.64:111
TCP TTL:26 TOS:0x0 ID:39426
**SF**** Seq: 0x3CEF17E6 Ack: 0x2F80FB0C Win: 0x404

[**] RPC Info Query [**]
08/14-17:03:21.011106 195.251.213.58:604 -> a.b.c.d:111
TCP TTL:48 TOS:0x0 ID:33616 DF
*****PA* Seq: 0xA94FCEFC Ack: 0x193EB5DA Win: 0x7D78
80 00 00 28 4A E4 1C 10 00 00 00 00 00 00 00 02 ...(J.....
00 01 86 A0 00 00 00 02 00 00 00 04 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 ....
```

1. Source of Trace:

http://www.sans.org/y2k/081700.htm

2. Detect Generated By:

The first part of this detect was generated by TCPDump. The second and third sections of the trace appear to be generated by the Snort Intrusion Detection System.

3. Probability Source Was Spoofed

Very low. Attacker is performing a targeted scan of network for machines running portmapper in order to find rpc (remote procedure call) services that will potentially be vulnerable to buffer overflow exploits. If the source address is spoofed, the attacker will be unable to receive the results of the scan and subsequent query.

4. Description of Attack

The attacker is performing a sequential scan against the "my.net" network in search of machines running the portmapper service (TCP/111). The portmapper service will return information on the rpc services running on the target. There are several known buffer overflow vulnerabilities that exist with these services, and the attacker is surely in search of vulnerable services that can be exploited. After conducting the network scan, the attacker finds at least one machine running portmapper, to which the attacker issues an rpcinfo request in order to retrieve the port #'s of the running rpc services. If this request is successful, the attacker can then attempt a buffer overflow exploit against the running services.

There are numerous CVE entries associated with the rpc services that could be vulnerable to a buffer overflow exploit:
rpc.ttdbserverd - CVE-1999-0687, CVE-1999-0003, CVE-1999-0693
rpc.cmsd – CVE-1999-0696
rpc.statd - CVE-1999-0018, CVE-1999-0019

1

5. Attack Mechanism

The attack begins with a sequential SYN-FIN scan of the target network. If the scanned machine is running the portmapper service (TCP/111) it will respond with an ACK. If the service is not running, a RESET will be issued to the attacker notifying him that the port is closed. The SYN-FIN combination is probably being used in an attempt to elude IDS or firewall systems that may only be filtering for SYN only connections.

Following the sequential scan, the attacker issues an rpcinfo request to at least one of the targeted machines. rpcinfo –p will contact the portmapper service and retrieve a listing of all of the RPC services running and their associated port numbers. This request is identified by the third section of the trace. The hexadecimal string "00 01 86 A0 00 00 00 02 00 00 00 04" within the content of the TCP packet trips the Snort detection rule identifying this as an RPC Info Query(see rule below.) Since the full trace is not presented on the GIAC website, it is unclear how much information was returned in the response to this request. The potential outcome is that the attacker has a listing of all of the RPC services and the ports they are running on.

Once the attacker has the port number of a potentially vulnerable RPC service, he will likely attempt to run one of the many available buffer overflow exploits (e.g. rpc.ttdbserver.c)

An automated tool is clearly being used to perform the scan and rpcinfo request seen in this detect. The SYN-FIN packets are obviously crafted since the SYN and FIN flags should never occur together. Additionally, the packets are being sent at essentially the same time indicating an automated process. To top it off, the rpcinfo request is occurring only 4 seconds after the last listed SYN-FIN scan indicating that it is also very likely incorporated into the automated tool.

Snort rule being tripped:
alert tcp !$HOME_NET any -> $HOME_NET 111 (msg:"RPC Info Query"; content:"|00 01 86 A0 00 00 00 02 00 00 00 04|";) alert

6. Correlations

Scans for RPC services are very commonplace. Although I found no other scans listed from this particular host, some other reports of scans listed on GIAC are:

http://www.sans.org/y2k/081600.htm
http://www.sans.org/y2k/081900.htm
http://www.sans.org/y2k/082100.htm

7. Evidence of Active Targeting

There does not appear to be active targeting involved here since a scan is performed of the entire target network. Specific machines within the network will be targeted if they are found to be running portmapper.

8. Severity

Since this detect was taken from the GIAC site, the full severity of this attack is unknown. As a result I will make some assumptions:

- There is a firewall in place that is blocking portmapper (TCP/111)
- Only the single machine listed in the trace responded to the SYN/FIN packets with a SYN/ACK. It lies in the DMZ. The machine is running the latest patched versions of the RPC services.
- That machine is a Web Server for the organization

1

Using the formula for determining severity [Criticality + Lethality – System Countermeasures – Network Countermeasures]:
5 (they system is a Web Server for the organization) + 5 (a buffer overflow would grant root access) – 4 (the system is running with the latest versions of RPC services (but it likely shouldn't be running them at all)) – 0 (machine in DMZ) = 6

This comes out as a pretty high severity attack. Only one machine is responding to the scan and it is running the latest versions of the rpc services so it shouldn't be vulnerable to existing rpc exploits. However, it probably should not be running rpc services in the first place since it is a critical machine (a web server) and since it is unprotected by the firewall (in the DMZ.) There is always the possibility that new RPC exploits will be uncovered (probably a good possibility), and running unnecessary services with this track record is inadvisable.

9. Defensive Recommendations
The rpc services should be removed from the responding machine. If some of the rpc services are required, they should be kept at the latest patch level to guard against any newly discovered exploits.

10. Multiple Choice Question (based on trace)
Which service runs on TCP port 111?
a. IMAP
b. POP-2
c. POP-3
d. Portmapper

Correct Answer: d

Detect #2 – NetBios Name Resolution

```
Jul 24 23:59:22 picard kernel: Packet log: input DENY wp1_fr16
  PROTO=17 165.91.71.63:137 204.x.x.1:137
  L=78 S=0x00 I=42319 F=0x0000 T=114 (#57)
Jul 24 23:59:23 picard kernel: Packet log: input DENY wp1_fr16
  PROTO=17 165.91.71.63:137 204.x.x.1:137
  L=78 S=0x00 I=44367 F=0x0000 T=114 (#57)
Jul 24 23:59:23 picard kernel: Packet log: input DENY wp1_fr16
  PROTO=17 165.91.71.63:137 204.x.x.1:137
  L=78 S=0x00 I=45903 F=0x0000 T=114 (#57)
Jul 24 23:59:23 picard kernel: Packet log: input DENY wp1_fr16
  PROTO=17 165.91.71.63:137 204.x.x.1:137
  L=78 S=0x00 I=46159 F=0x0000 T=114 (#57)
Jul 24 23:59:24 picard kernel: Packet log: input DENY wp1_fr16
  PROTO=17 165.91.71.63:137 204.x.x.1:137
  L=78 S=0x00 I=46415 F=0x0000 T=114 (#57)
Jul 24 23:59:24 picard kernel: Packet log: input DENY wp1_fr16
  PROTO=17 165.91.71.63:137 204.x.x.1:137
  L=78 S=0x00 I=47439 F=0x0000 T=114 (#57)
Jul 24 23:59:24 picard kernel: Packet log: input DENY wp1_fr16
  PROTO=17 165.91.71.63:137 204.x.x.1:137
  L=78 S=0x00 I=47951 F=0x0000 T=114 (#57)
```

1. Source of Trace

http://www.sans.org/y2k/072600.htm

2. Detect Generated By

This detect was generated by ipchains.

The fields in the trace are as follows:

Line 1: Jul 24 23:59:22 picard kernel: Packet log: input DENY wp1_fr16
(1) (2)

Line 2: PROTO=17 165.91.71.63:137 204.x.x.1:137
(3) (4) (5)

Line 3: L=78 S=0x00 I=42319 F=0x0000 T=114 (#57)
(6) (7) (8) (9) (10) (11)

(1) Timestamp
(2) Indicates interface packet came in on and what to do with packet (Deny)
(3) Protocol used
(4) Source address and port
(5) Destination address and port
(6) Size of packet
(7) Type of Service
(8) IP ID
(9) Fragmentation flags

(10)Time to live
(11)Rule number triggering detect

3. Probability Source Was Spoofed
This is a reconnaissance effort to enumerate information (such as users and groups) on the targeted machine. It is unlikely that the source address is spoofed because the attacker would not receive any results from the scan using a forged address.

4. Description of Attack
The attacker is performing a reconnaissance scan of the targeted machine in an effort to enumerate user and group information. Once the attacker has a list of users and groups, he will be able to employ password guessing to attempt to gain access to the target.

5. Attack Mechanism
The user is likely using NBTSTAT –A to enumerate the user and group information. NBTSTAT –A will issue a request from port 137 on the source machine to port 137 on the target machine and will return the contents of the target machine's NetBIOS name table.

6. Correlations
This is a common scan that has been reported numerous times on the GIAC site. It has been reported by the same individual on at least the following reports:

http://www.sans.org/y2k/081300.htm
http://www.sans.org/y2k/081800.htm
http://www.sans.org/y2k/082100.htm

7. Evidence of Active Targeting
According to the GIAC entry that this detect was taken from, this was the only machine targeted on the network. This could indicate that some previous reconnaissance has been done to choose this particular machine to attack.

8. Severity
Since this detect was taken from the GIAC site, the full severity of this attack is unknown. As a result I will make some assumptions:

- The targeted machine is an NT Server with NetBIOS Name Service running

Using the formula for determining severity [Criticality + Lethality – System Countermeasures – Network Countermeasures]:
4 (they system is a Server for the organization) + 2 (attacker can gain information on users) – 3 (unknown how secure system itself is) – 5 (The firewall blocks incoming request to NetBIOS Name Service) = -2

This is a low severity attack. Although this scan could provide the attacker with user and group information, the firewall effectively blocks the incoming packets and defeats the attack.

9. Defensive Recommendations
The firewall provides the most critical protection for this machine, and it is already in place. An additional measure to take would be to disallow null connections on the targeted machine.

1

10. Multiple Choice Question
Which of the following commands will enumerate user names and groups on a NT machine?
a. netstat
b. nbtstat
c. net view
d. net use

correct answer: b

Detect #3 – Bind Inverse Query/Version Query

Name: ncn.ca
Address: 216.123.177.82
Aug 19 14:26:47 hostname named[xxx]: unapproved query from [216.123.177.82].4974 for "version.bind"
[**] IDS277/named-probe-iquery [**]
08/19-14:26:47.496410 216.123.177.82:4974 -> a.b.c.2:53
UDP TTL:50 TOS:0x0 ID:32151
Len: 35
1F 8F 09 80 00 00 00 01 00 00 00 00 00 00 01 00 ....
01 00 00 7A 69 00 04 04 03 02 01 ...zi.....
[**] MISC-DNS-version-query [**]
08/19-14:26:47.878934 216.123.177.82:4974 -> a.b.c.2:53
UDP TTL:50 TOS:0x0 ID:32190
Len: 38
42 24 01 80 00 01 00 00 00 00 00 00 07 76 65 72 B$.........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03 sion.bind.....

1. Source of Trace
http://www.sans.org/y2k/082400-1030.htm

2. Detect Generated By
This detect was generated by Snort. In particular, the following two Snort rules triggered it:

alert udp !$HOME_NET any -> $HOME_NET 53 (msg:"IDS277 - NAMED Iquery Probe"; content: "|0980 0000 0001 0000 0000|"; offset: 2; depth: 16;)
alert udp !$HOME_NET any -> $HOME_NET 53 (msg:"MISC-DNS-version-query"; content:"version|04|bind|0000 1000 03";))

The fields in the trace are as follows:

Line1: [**] IDS277/named-probe-iquery [**]
(1)
line 2: 08/19-14:26:47.496410 216.123.177.82:4974 -> a.b.c.2:53
(2) (3) (4) (5) (6)
line3: UDP TTL:50 TOS:0x0 ID:32151
(7) (8) (9) (10)
line4: Len: 35

(11)
line 5-6: 1F 8F 09 80 00 00 00 01 00 00 00 00 00 00 01 00 ..... 01 00 00 7A 69 00 04 04 03 02 01 ...zi.....
(12)

(1) Snort Rule msg
(2) Timestamp
(3) Source address
(4) Source port
(5) Destination address
(6) Destination port
(7) Protocol used
(8) Time To Live for packet
(9) Type of Service
(10) IP ID
(11) Size of UDP packet (including 8 byte header)
(12) Content of UDP packet

3. Probability Source Was Spoofed
The attacker is attempting to determine the version of BIND being run on the target. If the attacker is using a spoofed address, then he would have no way of receiving the desired reconnaissance (e.g. the bind version). As a result, it is very unlikely that the source of the attack is spoofed.

4. Description of Attack
The attacker is attempting to determine the version of BIND being run on the target machine (run on UDP/53) and if it supports inverse queries in order to identify if the target machine will be susceptible to a buffer overflow exploit. There are many known buffer overflow vulnerabilities with earlier versions of BIND (pre 8.2.2 patch level 5) that will give an attacker root access.

This is CVE-1999-0009.

5. Attack Mechanism
The attacker first issues an inverse query request to determine if the target allows inverse queries. The attacker follows this with a regular query requesting the bind version (identifiable by the "version.bind" in the content of the second packet.) This is the same mechanism used by dig to retrieve the BIND version number (several other tools appear to perform this same request.) This was almost surely performed by an automated script given the close proximity in time that the inverse query and regular queries were performed. If a vulnerable target was found, the attacker would very likely have followed with a BIND buffer overflow exploit.

6. Correlations
This is a common scan that can be found in numerous previous GIAC reports. Examples are:
http://www.sans.org/y2k/080100.htm
http://www.sans.org/y2k/071600.htm

7. Evidence of Active Targeting
Since this detect was taken from the GIAC site, it is unknown whether or not this particular machine was actively targeted. If an assumption is made that the machine is part of a larger network and that it alone was singled out, then it is likely that some previous reconnaissance has been done such as network mapping.

1

8. Severity

Since this detect was taken from the GIAC site, the full severity of this attack is unknown. As a result I will make some assumptions:

- The system is a Unix Server that is not the DNS server for its organization
- The system is running a vulnerable version of BIND
- The system lies behind the organization's firewall
- The firewall blocks incoming requests to UDP/53.

Using the formula for determining severity [Criticality + Lethality – System Countermeasures – Network Countermeasures]:
4 (they system is a Server for the organization) + 5 (a buffer overflow would grant root access) – 0 (the system is running with a vulnerable version of BIND) – 5 (The firewall blocks incoming request to DNS) = 4

This is a moderate to high severity attack. Although it is unlikely that an attacker will be able to reach the machine through the firewall, there is no protection against attacks generated from the inside of the firewall.

9. Defensive Recommendations

Some countermeasures can be taken to help ensure the integrity of the system. Since the system is not being used as a DNS server (given the assumptions above), then the service should be disabled entirely to prevent any possible exploits of it. If there is some valid reason to continue to run DNS on the box, then the administrator should ensure that it is patched to the latest version of BIND.

10. Multiple Choice Question

08/19-14:26:47.878934 216.123.177.82:4974 -> a.b.c.2:53
UDP TTL:50 TOS:0x0 ID:32190
Len: 38
42 24 01 80 00 01 00 00 00 00 00 00 07 76 65 72 B$.....ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03 sion.bind.....

For the previous packet, which of the following is TRUE:
a. This is a BIND query
b. This is an Domain Inverse Query
c. This uses TCP
d. All of the above

Correct answer: a

Detect #4 – Trojan Scans

```
FWIN,2000/07/02,02:23:36 -5:00
GMT,64.228.193.42:3293,X.X.X.156:12345,TCP
FWIN,2000/07/02,02:23:48 -5:00
GMT,64.228.193.42:3448,X.X.X.156:12346,TCP
FWIN,2000/07/02,02:23:52 -5:00 GMT,64.228.193.42:3603,X.X.X.156:20034,TCP
FWIN,2000/07/02,02:23:58 -5:00
GMT,64.228.193.42:3758,X.X.X.156:31337,TCP
FWIN,2000/07/02,02:24:04 -5:00
GMT,64.228.193.42:3914,X.X.X.156:1243,TCP
```

1. Source of Trace

http://www.sans.org/y2k/070500.htm

2. Detect Generated By

This detect was generated by ZoneAlarm.

3. Probability Source Was Spoofed

The attacker is "trolling for trojans". In order to find a vulnerable system, the attacker must be able to receive the results of the scan. As a result, it is very unlikely that the attacker is spoofing his address.

4. Description of Attack

The attacker is scanning a single host in an attempt to find trojans that can be used to gain control of the target. In particular, the attacker is scanning for NetBus (TCP/12345, TCP/12346), NetBus 2 Pro(TCP/20034), Back Orifice (UDP 31337), SubSeven (TCP/1243). If the attacker receives a response to any of the scan queries, he will then use that particular trojan to gain control of the target.

5. Attack Mechanism

The attacker sends a series of packets to determine if any well known trojans are installed on the target machine and listening on their typical ports. If the attacker receives a response from any of the packets, he will know that the responding port has a trojan running on it that the attacker can then use to gain control over the target machine.

6. Correlations

Scanning for trojans is common practice on the internet. In reviewing the GIAC site, I did not find any attacks from the same source address, but there are a number of other similar scans for trojans. Examples are:

http://www.sans.org/y2k/072200.htm
http://www.sans.org/y2k/081500.htm

7. Evidence of Active Targeting

Since ZoneAlarm is being used in the detect, this is likely someone's personal machine or network. As a result, the attacker could very likely have been scanning the entire subnet around the target which would not pick up in its detect. The timeframe between the packet arrivals is somewhat spaced out (4 – 12 seconds), but this could be the product of a designed slow scan, or alternatively of a scan that is processing a subnet for a particular trojan and then moving onto the next trojan.

8. Severity
Since this detect was taken from the GIAC site, the full severity of this attack is unknown. As a result I will make some assumptions:

- The system is the user's personal box
- The system is running up-to date anti-virus software

Using the formula for determining severity [Criticality + Lethality – System Countermeasures – Network Countermeasures]:
4 (assuming the system is the user's personal box) + 5 (a trojan would grant root access) – 5 (the system has up-to date anti-virus software) – 5 (ZoneAlarm is blocking these packets) = -1

This is a low risk attack. The combination of ZoneAlarm and the up-to date anti-virus software (assumed) will effectively mitigate the risks involved.

9. Defensive Recommendations
Continue to update anti-virus definitions and the ZoneAlarm rules database.

10. Multiple Choice Question
What trojan is typically associated with the port TCP/1243?

a. SubSeven
b. Deep Throat
c. NetBus
d. None of the Above

Correct answer: a

Detect #5 – Scan for IMAP

```
Jun 25 21:03:27 dns1 portsentry[608]: attackalert:
  Connect from host: 203.239.106.61/203.239.106.61 to TCP port: 143
Jun 25 21:03:36 dns1 portsentry[608]: attackalert:
  Connect from host: 203.239.106.61/203.239.106.61 to TCP port: 143
Jun 25 21:03:27 dns3 portsentry[301]: attackalert:
  Connect from host: 203.239.106.61/203.239.106.61 to TCP port: 143
Jun 25 21:03:36 dns3 portsentry[301]: attackalert:
  Connect from host: 203.239.106.61/203.239.106.61 to TCP port: 143
```

1. Source of Trace
http://www.sans.org/y2k/062900.htm

2. Detect Generated By
This detect was generated by portsentry.

3. Probability Source Was Spoofed
The attacker is most likely attempting to identify if targets are running IMAP. In order to find a vulnerable system, the attacker must be able to receive the results of the scan. As a result, it is very unlikely that the attacker is spoofing his address.

4. Description of Attack
The attacker is attempting to connect to the IMAP service on the targets (DNS1, DNS3). The IMAP service has known buffer overflow vulnerabilities than can be exploited to give the attacker root access on the target machines.

The CVE # for this are:
CVE-1999-0005, CVE-1999-0042, CVE-1999-0920

5. Attack Mechanism
The attacker is sending packets to connect to the IMAP service on the targets. From the trace, it is not fully apparent the specific type of scan that is being performed (SYN-FIN, XMAS tree, etc…), but if a target sends back a RESET response, the attacker will know that IMAP is not running. If the attacker receives a SYN-ACK response, he will know that IMAP is available. Its not entirely clear why each of the targeted machines is receiving two packets, but one explanation could be two different scan types being made in order to circumvent filtering controls that might be in place. If the attacker is able to connect to the IMAP service, he will then attempt a buffer overflow exploit to gain root control of the box.

6. Correlations
http://www.sans.org/y2k/062100-1030.htm
http://www.sans.org/y2k/062500.htm
http://www.sans.org/y2k/062700-1200.htm

7. Evidence of Active Targeting
The two machines identified in this scan are listed as DNS1 and DNS3. Since the attack is directed at the DNS servers, there is a pretty strong indication that the attacker had some previous knowledge of the network prior to launching the scan.

8. Severity
Using the formula for determining severity [Criticality + Lethality – System Countermeasures – Network Countermeasures]:
5 (the targets are DNS servers) + 5 (an IMAP buffer overflow would grant root access) – 5 (PortSentry on the DNS servers will block attempts) – 1 (DNS servers sitting in DMZ will have little network protection) = 4

This is probably not a high risk attack. The use of PortSentry should effectively block the connection attempts to the IMAP service. Since the DNS services reside in the DMZ, there will be little in the way of additional network protection.

9. Defensive Recommendations
The DNS servers should be assessed to determine if they actually require IMAP to be running. It is likely that they do not need it. If it is required, it should be kept at the latest patch level and the PortSentry configuration should be maintained to prevent unauthorized connections.

10. Multiple Choice Question
Which port does IMAP typically run on?
a. TCP/143
b. UDP/143
c. TCP/161
d. UDP/161
e. Both a and b
f. Both c and d

Correct answer e

1

Assignment 2 – Evaluate an Attack

This is an analysis of exploiting unprotected file shares using the Microsoft "net view" and "net use" commands.

1. **Location of Attack**
   The net view and net use commands are standard parts of the NT and win 9x operating systems.

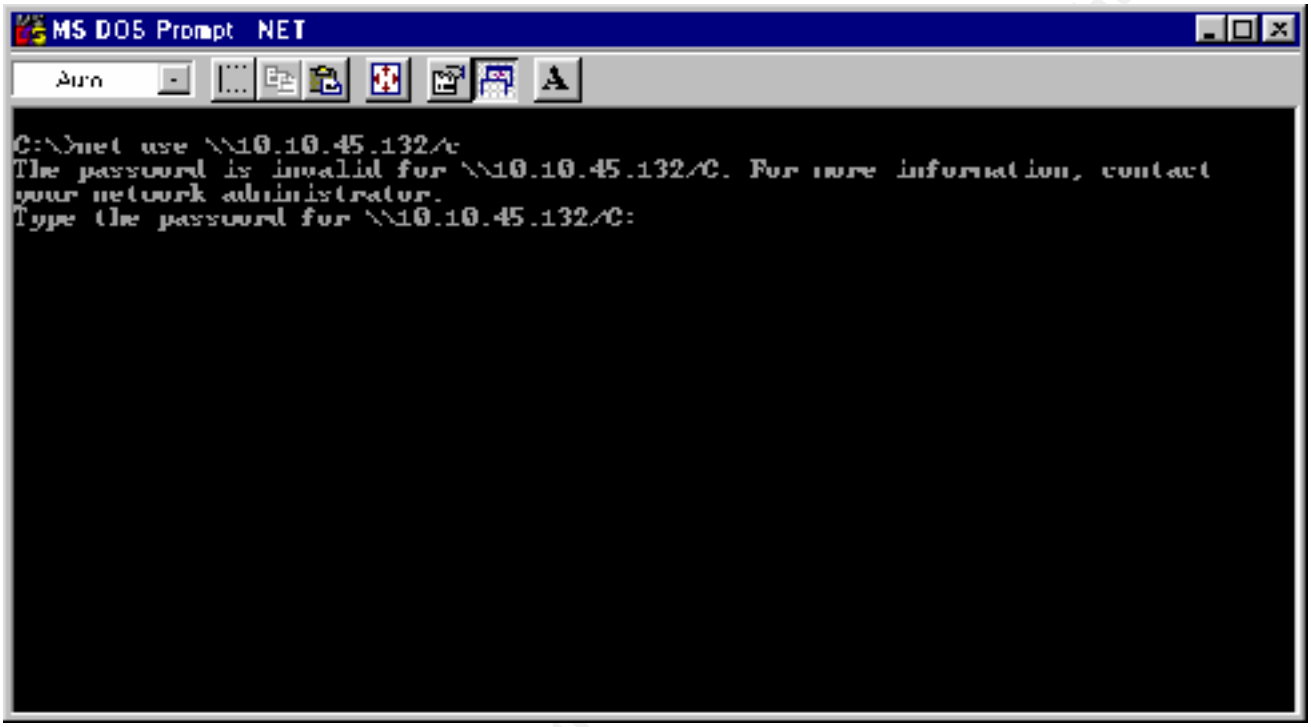
2. **Description of Attack**
   The Microsoft "net view" and "net use" commands are designed to enable discovery and sharing of resources between computers on a network. "net view" will display a list of computers in a specified domain or the shared resources available on a specific computer. "net use" can be used to then connect your computer to the shared resources on another computer.

   Although these are both very useful commands, they are also a simple tool for attackers to use to enumerate the computers in a domain or to discover open and unprotected file shares on a specific machine. The steps of the attack are:

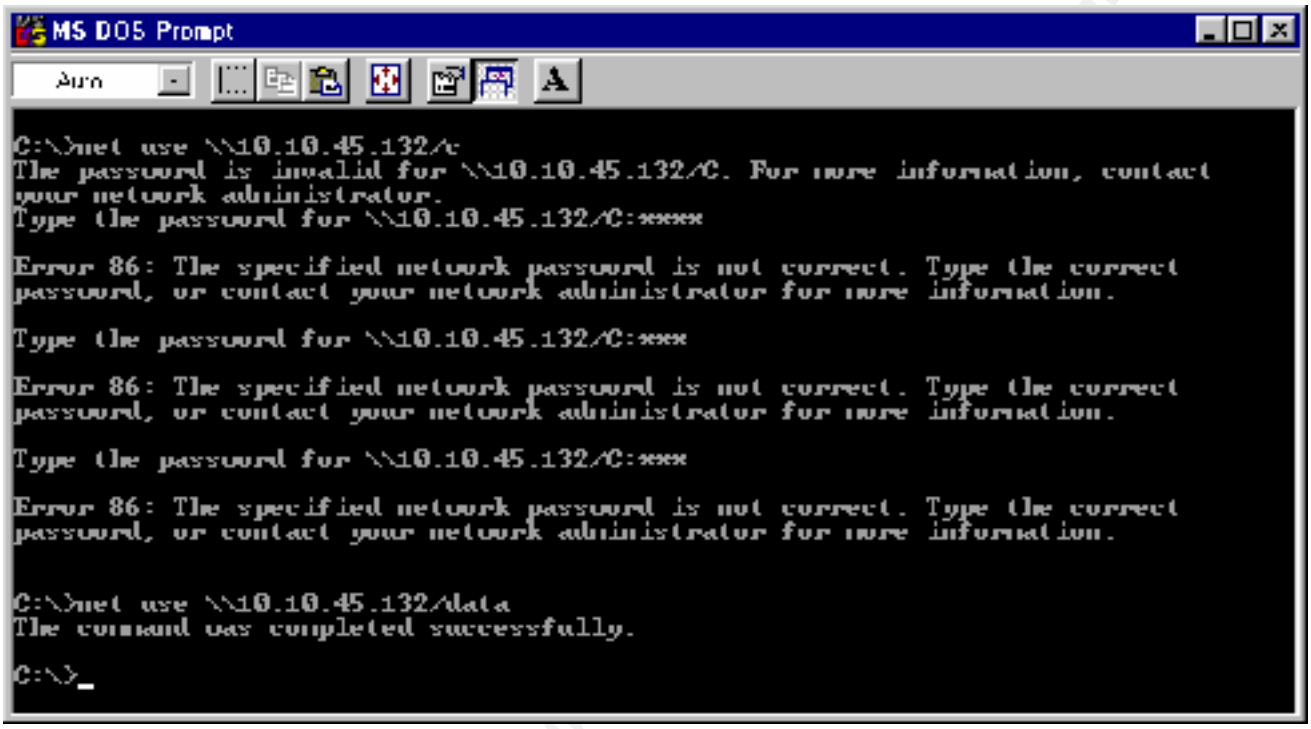
   a. The attacker performs a "net view \\10.10.45.132". This returns a list of all the open file shares on the target machine (see screen shot.) Note that there are three open file shares and one of them appears to be for the entire hard disk (C.)

1

b. The attacker now performs a "net use f: \\10.10.45.132/c". This will attempt to map a drive to the share "C" which will likely grant access to the entire remote hard disk. From the screen shot we can see that the share "C" is password protected. The attacker could attempt a brute-force effort to gain access, but it will be easier to try the other available shares to see if they are unprotected.



c. The attacker now issues a "net use \\10.10.45.132/data" command to attempt to connect to the "data" share. From the screenshot, we can see that this share is unprotected and does not require a password for connection. The attacker will now have access to any files within this share and can potentially place his own files (e.g. trojans) there as well.



3. Annotated Network Trace

Network traces of this attack were captured using a packet sniffer called Analyzer. First we will look at the trace for the "net view" command:

1 | 16h:29m:09s:756855us | FFFFFF-FFFFFF | 0010A4-06A79A | IPX: 0x00000000.0x0552 => 0x00000000.0x0551 | NETBIOS |
2 | 16h:29m:09s:773659us | FFFFFF-FFFFFF | 0010A4-06A79A | ARP: Hardware type= 1, Protocol Type = IP (0800h) | ARP Request |
3 | 16h:29m:09s:773958us | 0010A4-06A79A | 00E029-71C279 | ARP: Hardware type= 1, Protocol Type = IP (0800h) | ARP Reply |
4 | 16h:29m:09s:774038us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (78) | UDP: Port (137 => 137) NETBIOS NAME: Request: Query |
5 | 16h:29m:09s:774565us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (275) | UDP: Port (137 => 137) NETBIOS NAME: Response: Query |
6 | 16h:29m:09s:774716us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (64) | TCP: Port (1034 => 139) Data (SN 1994041, ACK 0) |
7 | 16h:29m:09s:774996us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (44) | TCP: Port (139 => 1034) Data (SN 5481384, ACK 1994042) |
8 | 16h:29m:09s:775126us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (40) | TCP: Port (1034 => 139) Data (SN 1994042, ACK 5481385) |
9 | 16h:29m:09s:775140us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (112) | TCP: Port (1034 => 139) Data (SN 1994042, ACK 5481385) NETBIOS SESSION: Session Request: |
10 | 16h:29m:09s:775542us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (44) | TCP: Port (139 => 1034) Data (SN 5481385, ACK 1994114) NETBIOS SESSION: Positive Session Response |
11 | 16h:29m:09s:776006us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (198) | TCP: Port (1034 => 139) Data (SN 1994114, ACK 5481389) NETBIOS SESSION: Session Message SMB: Negotiate: |
12 | 16h:29m:09s:776315us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (121) | TCP: Port (139 => 1034) Data (SN 5481389, ACK 1994272) NETBIOS SESSION: Session Message SMB: Negotiate: |

1

13 | 16h:29m:09s:777272us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (213) | TCP: Port (1034 => 139) Data (SN 1994272, ACK 5481470) NETBIOS SESSION: Session Message SMB: Session Setup andX: |
14 | 16h:29m:09s:777531us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (96) | TCP: Port (139 => 1034) Data (SN 5481470, ACK 1994445) NETBIOS SESSION: Session Message SMB: Session Setup andX: |
15 | 16h:29m:09s:777956us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (139) | TCP: Port (1034 => 139) Data (SN 1994445, ACK 5481526) NETBIOS SESSION: Session Message SMB: Transaction: |
16 | 16h:29m:09s:778404us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (226) | TCP: Port (139 => 1034) Data (SN 5481526, ACK 1994544) NETBIOS SESSION: Session Message SMB: Transaction: |
17 | 16h:29m:09s:892641us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (40) | TCP: Port (1034 => 139) Data (SN 1994544, ACK 5481712) |
18 | 16h:29m:12s:027265us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (79) | TCP: Port (1034 => 139) Data (SN 1994544, ACK 5481712) NETBIOS SESSION: Session Message SMB: Tree Disconnect: |
19 | 16h:29m:12s:027531us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (79) | TCP: Port (139 => 1034) Data (SN 5481712, ACK 1994583) NETBIOS SESSION: Session Message SMB: Tree Disconnect: |
20 | 16h:29m:12s:027809us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (40) | TCP: Port (1034 => 139) Data (SN 1994583, ACK 5481751) |
21 | 16h:29m:12s:028005us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (40) | TCP: Port (139 => 1034) Data (SN 5481751, ACK 1994584) NETBIOS SESSION: |
22 | 16h:29m:12s:028113us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (40) | TCP: Port (1034 => 139) Data (SN 1994584, ACK 5481752) |

On lines 4 and 5 we see an initial connection being made on UDP 137. The source machine is retrieving the NETBIOS name information to use in its request for share information.

4 | 16h:29m:09s:774038us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (78) | UDP: Port (137 => 137) NETBIOS NAME: Request: Query |
5 | 16h:29m:09s:774565us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (275) | UDP: Port (137 => 137) NETBIOS NAME: Response: Query |

On lines 6 through 8 we see the TCP handshake to establish the connection on port 139.

6 | 16h:29m:09s:774716us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (64) | TCP: Port (1034 => 139) Data (SN 1994041, ACK 0) |
7 | 16h:29m:09s:774996us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (44) | TCP: Port (139 => 1034) Data (SN 5481384, ACK 1994042) |
8 | 16h:29m:09s:775126us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (40) | TCP: Port (1034 => 139) Data (SN 1994042, ACK 5481385) |

Lines 9 – 13 are establishing the SMB connection on the hidden interprocess communications share (IPC$). If we look at the content of the packet on line 13, we see the specific connection request to that share being made (in bold.)

* 00 E0 29 71 | C2 79 00 10 | A4 06 A7 9A | 08 00 45 0E [..)q.y.....E.]
* 00 D5 D3 00 | 40 00 80 06 | B7 F7 0A 0A | 2D 85 0A 0A [....@.....-...]
* 2D 84 04 0A | 00 8B 00 1E | 6E 20 00 53 | A3 FE 50 18 [-.....n .S..P.]
* 21 E3 97 A5 | 00 00 00 00 | 00 A9 FF 53 | 4D 42 73 00 [!.....SMBs.]
* 00 00 00 10 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 [.....]
* 00 00 00 00 | 00 43 00 01 | 00 01 0D 75 | 00 70 00 68 [.....C.....u.p.h]
* 0B 02 00 00 | 00 0D 00 00 | 80 01 00 01 | 00 00 00 00 [.....]
* 00 01 00 00 | 00 33 00 00 | 00 45 52 49 | 43 5F 54 5F [.....3...ERIC_T_]
* 42 52 45 4C | 53 46 4F 52 | 44 00 47 4D | 55 2D 4C 4D [BRELSFORD.GMU-LM]
* 55 00 57 69 | 6E 64 6F 77 | 73 20 34 2E | 30 00 57 69 [U.Windows 4.0.Wi]
* 6E 64 6F 77 | 73 20 34 2E | 30 00 04 FF | 00 00 00 02 [ndows 4.0.....]
* 00 18 00 2E | 00 59 9F 7C | F9 E8 94 BC | B1 B4 3E 78 [.....Y.|.....>x]
* EB 68 FA 4E | C6 0D 15 44 | 71 53 06 EA | 21 5C 5C 42 [.h.N...DqS..!**B**]
* 52 45 4C 53 | 46 4F 52 44 | 31 5C 49 50 | 43 24 00 49 [**RELSFORD1\IPC$.I**]
* 50 43 00 | | | [**PC.**]

Lines 14 – 16 consist of the actual request for the share information, the response back from the target, and the acknowledgment from the source machine:

14 | 16h:29m:09s:777531us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (96) | TCP: Port (139 => 1034) Data (SN 5481470, ACK 1994445) NETBIOS SESSION: Session Message SMB: Session Setup andX: |
15 | 16h:29m:09s:777956us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (139) | TCP: Port (1034 => 139) Data (SN 1994445, ACK 5481526) NETBIOS SESSION: Session Message SMB: Transaction: |
16 | 16h:29m:09s:778404us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (226) | TCP: Port (139 => 1034) Data (SN 5481526, ACK 1994544) NETBIOS SESSION: Session Message SMB: Transaction: |

The content of the packet from line 15 shows the share information that is being returned:

* 00 10 A4 06 | A7 9A 00 E0 | 29 71 C2 79 | 08 00 45 00 [.....)q.y..E.]
* 00 E2 1E 01 | 40 00 20 06 | CC F8 0A 0A | 2D 84 0A 0A [....@. .....-...]
* 2D 85 00 8B | 04 0A 00 53 | A4 36 00 1E | 6F 30 50 18 [-.....S.6..o0P.]
* 20 42 2B 34 | 00 00 00 00 | 00 B6 FF 53 | 4D 42 25 00 [B+4.....SMB%.]
* 00 00 00 80 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 [................]
* 00 00 00 00 | 00 43 00 00 | 01 01 0A 00 | 08 00 76 00 [.....**C**.....v.]
* 00 00 08 00 | 37 00 00 00 | 76 00 40 00 | 00 00 00 7F [....7...v.@....]
* 00 00 00 8A | 4F 04 00 04 | 00 5C 44 41 | 54 41 00 00 [....O....**DATA**..]
* 00 00 00 00 | 00 00 00 00 | 00 00 FF 4F | A5 C8 43 00 [...........O..C.]
* 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 FE 4F [...............O]
* A5 C8 4D 59 | 20 44 4F 43 | 55 4D 45 4E | 54 53 00 00 [..**MY DOCUMENTS**..]
* 00 00 FD 4F | A5 C8 49 50 | 43 24 00 00 | 00 00 00 00 [...O..IPC$.....]
* 00 00 00 00 | 03 00 DA 4F | A5 C8 52 65 | 6D 6F 74 65 [.......O..Remote]
* 20 49 6E 74 | 65 72 20 50 | 72 6F 63 65 | 73 73 20 43 [Inter Process C]
* 6F 6D 6D 75 | 6E 69 63 61 | 74 69 6F 6E | 00 00 00 00 [ommunication....]

Next we will look at the trace from the "net use" command. For brevity, we will just look at the second net use command issued – "net use f: \\10.10.45.132/data":

1 | 17h:02m:19s:662744us | FFFFFF-FFFFFF | 0010A4-06A79A | IPX: 0x00000000.0x0552 => 0x00000000.0x0551 | NETBIOS |
2 | 17h:02m:19s:679743us | FFFFFF-FFFFFF | 0010A4-06A79A | ARP: Hardware type= 1, Protocol Type = IP (0800h) | ARP Request |
3 | 17h:02m:19s:679999us | 0010A4-06A79A | 00E029-71C279 | ARP: Hardware type= 1, Protocol Type = IP (0800h) | ARP Reply |
4 | 17h:02m:19s:680071us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (78) | UDP: Port (137 => 137) NETBIOS NAME: Request: Query |
5 | 17h:02m:19s:680601us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (275) | UDP: Port (137 => 137) NETBIOS NAME: Response: Query |
6 | 17h:02m:19s:680767us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (64) | TCP: Port (1039 => 139) Data (SN 3984238, ACK 0) |
7 | 17h:02m:19s:681089us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (44) | TCP: Port (139 => 1039) Data (SN 7471566, ACK 3984239) |
8 | 17h:02m:19s:681211us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (40) | TCP: Port (1039 => 139) Data (SN 3984239, ACK 7471567) |
9 | 17h:02m:19s:681228us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (112) | TCP: Port (1039 => 139) Data (SN 3984239, ACK 7471567) NETBIOS SESSION: Session Request: |
10 | 17h:02m:19s:681638us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (44) | TCP: Port (139 => 1039) Data (SN 7471567, ACK 3984311) NETBIOS SESSION: Positive Session Response |
11 | 17h:02m:19s:684094us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (78) | UDP: Port (137 => 137) NETBIOS NAME: Request: Query |
12 | 17h:02m:19s:684430us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (275) | UDP: Port (137 => 137) NETBIOS NAME: Response: Query |
13 | 17h:02m:19s:685025us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (198) | TCP: Port (1039 => 139) Data (SN 3984311, ACK 7471571) NETBIOS SESSION: Session Message SMB: Negotiate: |
14 | 17h:02m:19s:685380us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (121) | TCP: Port (139 => 1039) Data (SN 7471571, ACK 3984469) NETBIOS SESSION: Session Message SMB: Negotiate: |
15 | 17h:02m:19s:686411us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (212) | TCP: Port (1039 => 139) Data (SN 3984469, ACK 7471652) NETBIOS SESSION: Session Message SMB: Session Setup andX: |
16 | 17h:02m:19s:686673us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (95) | TCP: Port (139 => 1039) Data (SN 7471652, ACK 3984641) NETBIOS SESSION: Session Message SMB: Session Setup andX: |
17 | 17h:02m:19s:807521us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (40) | TCP: Port (1039 => 139) Data (SN 3984641, ACK 7471707) |

1

18 | 17h:02m:22s:248237us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (122) | TCP: Port (1039 => 139) Data (SN 3984641, ACK 7471707) NETBIOS SESSION: Session Message SMB: Transaction2: |
19 | 17h:02m:22s:248708us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (122) | TCP: Port (139 => 1039) Data (SN 7471707, ACK 3984723) NETBIOS SESSION: Session Message SMB: Transaction2: |
20 | 17h:02m:22s:374540us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (40) | TCP: Port (1039 => 139) Data (SN 3984723, ACK 7471789) |

As with the "Net View" command, lines 6 – 8 are establishing the TCP handshake on port 139:

6 | 17h:02m:19s:680767us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (64) | TCP: Port (1039 => 139) Data (SN 3984238, ACK 0) |
7 | 17h:02m:19s:681089us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (44) | TCP: Port (139 => 1039) Data (SN 7471566, ACK 3984239) |
8 | 17h:02m:19s:681211us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (40) | TCP: Port (1039 => 139) Data (SN 3984239, ACK 7471567) |

Lines 15 – 17 contain the connection request to the "data" share. Line 15 contains the request, line 16 is the response, and line 17 is the source's acknowledgement of the response:

15 | 17h:02m:19s:686411us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (212) | TCP: Port (1039 => 139) Data (SN 3984469, ACK 7471652) NETBIOS SESSION: Session Message SMB: Session Setup andX: |
16 | 17h:02m:19s:686673us | 0010A4-06A79A | 00E029-71C279 | IP: 10.10.45.132 => 10.10.45.133 (95) | TCP: Port (139 => 1039) Data (SN 7471652, ACK 3984641) NETBIOS SESSION: Session Message SMB: Session Setup andX: |
17 | 17h:02m:19s:807521us | 00E029-71C279 | 0010A4-06A79A | IP: 10.10.45.133 => 10.10.45.132 (40) | TCP: Port (1039 => 139) Data (SN 3984641, ACK 7471707) |

The contents of the packet on line 15 show the actual connection request to the "data" share:

* 00 E0 29 71 | C2 79 00 10 | A4 06 A7 9A | 08 00 45 0A [..)q.y.....E.]
* 00 D4 2F 01 | 40 00 80 06 | 5B FC 0A 0A | 2D 85 0A 0A [../.@...[...-...]
* 2D 84 04 0F | 00 8B 00 3C | CC 55 00 72 | 02 24 50 18 [-.....<.U.r.$P.]
* 21 E3 59 0F | 00 00 00 00 | 00 A8 FF 53 | 4D 42 73 00 [!.Y.....SMBs.]
* 00 00 00 10 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 [................]
* 00 00 00 00 | 00 43 00 01 | 02 01 0D 75 | 00 70 00 68 [.....C.....u.p.h]
* 0B 02 00 00 | 00 12 00 01 | 80 01 00 01 | 00 00 00 00 [................]
* 00 01 00 00 | 00 33 00 00 | 00 45 52 49 | 43 5F 54 5F [.....3...ERIC_T_]
* 42 52 45 4C | 53 46 4F 52 | 44 00 47 4D | 55 2D 4C 4D [BRELSFORD.GMU-LM]
* 55 00 57 69 | 6E 64 6F 77 | 73 20 34 2E | 30 00 57 69 [U.Windows 4.0.Wi]
* 6E 64 6F 77 | 73 20 34 2E | 30 00 04 FF | 00 00 00 02 [ndows 4.0.....]
* 00 18 00 2D | 00 F3 BF 95 | 0A 59 FC 39 | 28 1A 43 EA [...-.....Y.9(.C.]
* B6 87 39 04 | 89 FA 5B FF | FF 0E EA 5B | 13 5C 5C 42 [..9...[....[.**B**]
* 52 45 4C 53 | 46 4F 52 44 | 31 5C 44 41 | 54 41 00 41 [**RELSFORD1\DATA.A**]
* 3A 00 | | | [:.]

Assignment 3 - "Analyze This" Scenario

As part of our bid to provide security services to this facility, this organization has been asked to provide an analysis report of the IDS detection logs over the past 30 days. As part of this analysis, we intend to identify hostile traffic and any steps that can be taken to mitigate the associated risk.

Note: For clarity, the log files have been parsed to group related traffic signatures together.

1. WinGate proxy server connections

The largest volume of traffic on this network is destined for WinGate proxy servers. The proxy server will enable an individual to connect through it to surf the internet. As a result, the individual will appear to be coming from the proxy server's address and not their own. This is a popular feature for people that seek anonymity online, but it can also help to hide the identify of an attacker.

The following table summarizes the most prevalent source and destination addresses within the log files:

Source Address	# of Entries	Destination Address	# of Entries
202.38.128.188	22338	MY.NET.253.105	16045
MY.NET.253.12	6448	MY.NET.99.85	1090
128.231.171.123	2974	MY.NET.60.11	235
24.3.26.53	2010	MY.NET.100.59	108
136.160.4.159	1198	MY.NET.60.8	87
24.13.120.49	934	MY.NET.99.51	82
64.67.15.98	628		
199.128.230.169	619		
136.160.5.145	568		
63.27.117.212	562		

From the figures above and the patterns of connection requests, it is apparent that at least two computers within this network are running WinGate proxy servers. In particular:

- MY.NET.253.105:8080
- MY.NET.99.85:8080

The traces indicate that connections are actually being established and proxying is taking place given the repeated number of connection requests with the same source and destination addresses. Also indicative of this is the incrementing port number of the source address and the time interval over which the connections are taking place. The below excerpt from the logs illustrates this point:

```
05/24-14:26:32.568556  [**] WinGate 8080 Attempt [**] 136.160.4.159:1976 -> MY.NET.253.105:8080
05/24-14:26:33.564233  [**] WinGate 8080 Attempt [**] 136.160.4.159:1982 -> MY.NET.253.105:8080
05/24-14:26:35.066971  [**] WinGate 8080 Attempt [**] 136.160.4.159:1989 -> MY.NET.253.105:8080
05/24-14:26:35.627300  [**] WinGate 8080 Attempt [**] 136.160.4.159:1992 -> MY.NET.253.105:8080
05/24-14:26:36.490496  [**] WinGate 8080 Attempt [**] 136.160.4.159:1999 -> MY.NET.253.105:8080
05/24-14:26:36.807356  [**] WinGate 8080 Attempt [**] 136.160.4.159:2002 -> MY.NET.253.105:8080
05/24-14:26:40.783081  [**] WinGate 8080 Attempt [**] 136.160.4.159:2032 -> MY.NET.253.105:8080
05/24-14:26:41.065141  [**] WinGate 8080 Attempt [**] 136.160.4.159:2034 -> MY.NET.253.105:8080
05/24-14:26:41.377991  [**] WinGate 8080 Attempt [**] 136.160.4.159:2036 -> MY.NET.253.105:8080
05/24-14:26:41.486061  [**] WinGate 8080 Attempt [**] 136.160.4.159:2037 -> MY.NET.253.105:8080
05/24-14:26:42.132882  [**] WinGate 8080 Attempt [**] 136.160.4.159:2041 -> MY.NET.253.105:8080
*
*
```

```
*
05/24-14:38:20.566315  [**] WinGate 8080 Attempt [**] 136.160.4.159:1634 -> MY.NET.253.105:8080
05/24-14:38:24.778626  [**] WinGate 8080 Attempt [**] 136.160.4.159:1635 -> MY.NET.253.105:8080
05/24-14:40:15.164195  [**] WinGate 8080 Attempt [**] 136.160.4.159:1649 -> MY.NET.253.105:8080
```

In addition to legitimate proxying activities, there is also clear evidence of active scanning of the host network for WinGate proxy servers. A host with the source address 202.38.128.188 performed a sequential scan of the entire MY.NET network within a 39 minute interval.

```
06/01-01:59:13.012322  [**] WinGate 8080 Attempt [**] 202.38.128.188:4953 -> MY.NET.1.0:8080
06/01-01:59:13.036870  [**] WinGate 8080 Attempt [**] 202.38.128.188:4954 -> MY.NET.1.1:8080
06/01-01:59:13.049802  [**] WinGate 8080 Attempt [**] 202.38.128.188:4955 -> MY.NET.1.2:8080
06/01-01:59:13.100513  [**] WinGate 8080 Attempt [**] 202.38.128.188:4956 -> MY.NET.1.3:8080
06/01-01:59:13.181110  [**] WinGate 8080 Attempt [**] 202.38.128.188:4959 -> MY.NET.1.6:8080
06/01-01:59:13.216122  [**] WinGate 8080 Attempt [**] 202.38.128.188:4960 -> MY.NET.1.7:8080
06/01-01:59:13.340451  [**] WinGate 8080 Attempt [**] 202.38.128.188:4963 -> MY.NET.1.10:8080
06/01-01:59:13.351379  [**] WinGate 8080 Attempt [**] 202.38.128.188:4964 -> MY.NET.1.11:8080
06/01-01:59:13.584223  [**] WinGate 8080 Attempt [**] 202.38.128.188:4970 -> MY.NET.1.17:8080
06/01-01:59:13.619683  [**] WinGate 8080 Attempt [**] 202.38.128.188:4971 -> MY.NET.1.18:8080
06/01-01:59:13.663364  [**] WinGate 8080 Attempt [**] 202.38.128.188:4972 -> MY.NET.1.19:8080
06/01-01:59:13.684310  [**] WinGate 8080 Attempt [**] 202.38.128.188:4973 -> MY.NET.1.20:8080
06/01-01:59:13.717973  [**] WinGate 8080 Attempt [**] 202.38.128.188:4974 -> MY.NET.1.21:8080
*
*
*
06/01-02:38:26.736583  [**] WinGate 8080 Attempt [**] 202.38.128.188:2398 -> MY.NET.254.250:8080
06/01-02:38:26.753044  [**] WinGate 8080 Attempt [**] 202.38.128.188:2399 -> MY.NET.254.251:8080
06/01-02:38:27.028567  [**] WinGate 8080 Attempt [**] 202.38.128.188:2400 -> MY.NET.254.252:8080
06/01-02:38:27.042807  [**] WinGate 8080 Attempt [**] 202.38.128.188:2401 -> MY.NET.254.253:8080
06/01-02:38:27.045056  [**] WinGate 8080 Attempt [**] 202.38.128.188:2402 -> MY.NET.254.254:8080
06/01-02:38:27.049122  [**] WinGate 8080 Attempt [**] 202.38.128.188:2403 -> MY.NET.254.255:8080
```

There is also evidence that the internal host MY.NET.253.12 has been compromised. This host is performing a sequential scan of the .16, .19, .101, and .102 subnets. This scan is occurring between 5/28 and 6/1. However, there is a gap in the logging on 5/30 so additional subnets likely have been scanned as well.

The scan itself follows an interesting pattern:

- Two connection requests are made to port 1080 on the destination host always using source port numbers 43746 and 43747.
- Four connection requests are then made to port 8080 using source port numbers 43746, 43747, 43749, 43750
- Two additional requests are then made to port 1080 using source port numbers 43749 and 43750

```
05/29-04:59:00.935197  [**] WinGate 1080 Attempt [**] MY.NET.253.12:43746 -> MY.NET.16.234:1080
05/29-04:59:01.416507  [**] WinGate 1080 Attempt [**] MY.NET.253.12:43747 -> MY.NET.16.234:1080
05/29-04:59:33.623965  [**] WinGate 8080 Attempt [**] MY.NET.253.12:43746 -> MY.NET.16.234:8080
05/29-04:59:34.103378  [**] WinGate 8080 Attempt [**] MY.NET.253.12:43747 -> MY.NET.16.234:8080
05/29-05:01:34.304755  [**] WinGate 8080 Attempt [**] MY.NET.253.12:43749 -> MY.NET.16.234:8080
05/29-05:01:34.785441  [**] WinGate 8080 Attempt [**] MY.NET.253.12:43750 -> MY.NET.16.234:8080
05/29-05:02:10.574773  [**] WinGate 1080 Attempt [**] MY.NET.253.12:43749 -> MY.NET.16.234:1080
05/29-05:02:11.094899  [**] WinGate 1080 Attempt [**] MY.NET.253.12:43750 -> MY.NET.16.234:1080

05/31-22:27:21.156888  [**] WinGate 1080 Attempt [**] MY.NET.253.12:43746 -> MY.NET.101.95:1080
05/31-22:27:21.477444  [**] WinGate 1080 Attempt [**] MY.NET.253.12:43747 -> MY.NET.101.95:1080
05/31-22:27:43.151467  [**] WinGate 8080 Attempt [**] MY.NET.253.12:43746 -> MY.NET.101.95:8080
05/31-22:27:43.439376  [**] WinGate 8080 Attempt [**] MY.NET.253.12:43747 -> MY.NET.101.95:8080
05/31-22:29:05.668525  [**] WinGate 8080 Attempt [**] MY.NET.253.12:43749 -> MY.NET.101.95:8080
05/31-22:29:05.988126  [**] WinGate 8080 Attempt [**] MY.NET.253.12:43750 -> MY.NET.101.95:8080
05/31-22:29:27.180431  [**] WinGate 1080 Attempt [**] MY.NET.253.12:43749 -> MY.NET.101.95:1080
05/31-22:29:27.507107  [**] WinGate 1080 Attempt [**] MY.NET.253.12:43750 -> MY.NET.101.95:1080
```

2. RPC High Port Access

There are a significant number of "RPC High Port Access" entries in the Snort logs. The following table summarizes the most prevalent source and destination addresses within the log files:

Source Address	# of Entries	Destination Address	# of Entries
MY.NET.253.12	3249	MY.NET.217.2	1496
205.188.153.106	1976	MY.NET.16.169	8
205.188.153.100	1496	MY.NET.16.171	8
63.90.234.50	580	MY.NET.16.173	8
207.25.253.26	100	MY.NET.16.175	8

A large number of these are false positives. ICQ traffic from source port 4000 going to destination port 32771 is triggering the Snort RPC rule. ICQ servers typically operate on port 4000 and pick a high numbered port to connect with.

```
06/12-09:43:54.351183  [**] Attempted Sun RPC high port access [**] 205.188.153.106:4000 ->
MY.NET.218.66:32771
06/12-09:45:31.294379  [**] Attempted Sun RPC high port access [**] 205.188.153.106:4000 ->
MY.NET.218.66:32771
06/12-09:45:54.292615  [**] Attempted Sun RPC high port access [**] 205.188.153.106:4000 ->
MY.NET.218.66:32771
06/12-09:50:54.130382  [**] Attempted Sun RPC high port access [**] 205.188.153.106:4000 ->
MY.NET.218.66:32771
06/12-09:51:54.070618  [**] Attempted Sun RPC high port access [**] 205.188.153.106:4000 ->
MY.NET.218.66:32771
06/12-09:53:53.993889  [**] Attempted Sun RPC high port access [**] 205.188.153.106:4000 ->
MY.NET.218.66:32771
06/12-09:56:53.860173  [**] Attempted Sun RPC high port access [**] 205.188.153.106:4000 ->
MY.NET.218.66:32771
06/12-09:57:53.873593  [**] Attempted Sun RPC high port access [**] 205.188.153.106:4000 ->
MY.NET.218.66:32771
06/12-09:58:53.788880  [**] Attempted Sun RPC high port access [**] 205.188.153.106:4000 ->
MY.NET.218.66:32771
```

There are several source addresses that are using the port of 4000 and each of them resolves to an AOL ICQ server when an nslookup is performed.

The remaining RPC-related snort entries should raise some concern. The majority of the connection attempts are being made to port 32771. Under Solaris, the Rpcbind service listens on port 32771 in addition to the standard port 111. It is very likely that the attackers are attempting to connect to this service in order to find out what RPC services are being offered. There are several known buffer overflow vulnerabilities with RPC services that can be exploited to grant root access.

```
06/13-10:46:39.512793  [**] SUNRPC highport access! [**] 207.25.253.26:20 -> MY.NET.70.127:32771
06/13-10:46:39.619936  [**] SUNRPC highport access! [**] 207.25.253.26:20 -> MY.NET.70.127:32771
06/13-10:46:39.619987  [**] SUNRPC highport access! [**] 207.25.253.26:20 -> MY.NET.70.127:32771
06/13-10:46:39.620308  [**] SUNRPC highport access! [**] 207.25.253.26:20 -> MY.NET.70.127:32771
06/13-10:46:39.719026  [**] SUNRPC highport access! [**] 207.25.253.26:20 -> MY.NET.70.127:32771
06/13-10:46:39.724895  [**] SUNRPC highport access! [**] 207.25.253.26:20 -> MY.NET.70.127:32771

06/16-09:50:49.140278  [**] SUNRPC highport access! [**] 208.226.167.19:21 -> MY.NET.143.87:32771
06/16-09:50:55.704322  [**] SUNRPC highport access! [**] 208.226.167.19:21 -> MY.NET.143.87:32771
06/16-09:51:08.868385  [**] SUNRPC highport access! [**] 208.226.167.19:21 -> MY.NET.143.87:32771
06/16-09:51:14.620319  [**] SUNRPC highport access! [**] 208.226.167.19:21 -> MY.NET.143.87:32771
06/16-09:51:31.516747  [**] SUNRPC highport access! [**] 208.226.167.19:21 -> MY.NET.143.87:32771
06/16-09:51:34.786129  [**] SUNRPC highport access! [**] 208.226.167.19:21 -> MY.NET.143.87:32771
06/16-09:51:38.876762  [**] SUNRPC highport access! [**] 208.226.167.19:21 -> MY.NET.143.87:32771
06/16-09:51:43.242912  [**] SUNRPC highport access! [**] 208.226.167.19:21 -> MY.NET.143.87:32771
06/16-09:51:44.299559  [**] SUNRPC highport access! [**] 208.226.167.19:21 -> MY.NET.143.87:32771
```

The compromised host MY.NET.253.12 is also performing a scan to gather information about available RPC services. Note that the tool performing this scan in conjunction with the WinGate scan described above is utilizing the same port numbers for the different types of scans.

```
05/28-14:30:50.876461  [**] SUNRPC highport access! [**] MY.NET.253.12:43746 -> MY.NET.16.0:32771
05/28-14:30:51.185774  [**] SUNRPC highport access! [**] MY.NET.253.12:43747 -> MY.NET.16.0:32771
05/28-14:31:04.905230  [**] SUNRPC highport access! [**] MY.NET.253.12:43749 -> MY.NET.16.0:32771
05/28-14:31:05.245775  [**] SUNRPC highport access! [**] MY.NET.253.12:43750 -> MY.NET.16.0:32771
05/28-14:34:34.562778  [**] SUNRPC highport access! [**] MY.NET.253.12:43746 -> MY.NET.16.3:32771
05/28-14:34:34.860094  [**] SUNRPC highport access! [**] MY.NET.253.12:43747 -> MY.NET.16.3:32771
05/28-14:34:48.011149  [**] SUNRPC highport access! [**] MY.NET.253.12:43749 -> MY.NET.16.3:32771
05/28-14:34:48.331077  [**] SUNRPC highport access! [**] MY.NET.253.12:43750 -> MY.NET.16.3:32771
05/28-14:38:06.627119  [**] SUNRPC highport access! [**] MY.NET.253.12:43746 -> MY.NET.16.4:32771
05/28-14:38:06.945139  [**] SUNRPC highport access! [**] MY.NET.253.12:43747 -> MY.NET.16.4:32771
05/28-14:38:20.386266  [**] SUNRPC highport access! [**] MY.NET.253.12:43749 -> MY.NET.16.4:32771
05/28-14:38:20.698481  [**] SUNRPC highport access! [**] MY.NET.253.12:43750 -> MY.NET.16.4:32771
05/28-14:41:26.069867  [**] SUNRPC highport access! [**] MY.NET.253.12:43746 -> MY.NET.16.5:32771
05/28-14:41:26.399063  [**] SUNRPC highport access! [**] MY.NET.253.12:43747 -> MY.NET.16.5:32771
```

3. SNMP Public Access

SNMP activity is also being captured within the Snort logs. The following table summarizes the most prevalent source and destination addresses:

Source Address	# of Entries	Destination Address	# of Entries
MY.NET.97.183	418	MY.NET.101.192	1242
MY.NET.97.52	120	(*no other destinations)	
MY.NET.97.133	113		
MY.NET.97.12	93		
MY.NET.97.226	80		

It is very likely that this is simply normal network traffic that is showing up as a false positive. The strongest evidence for this is that all of the source addresses emanate from the MY.NET.97 subnet and are all destined for MY.NET.101.192. There are no external source addresses attempting SNMP connections and none of the connection attempts are from the known compromised system (MY.NET.253.12.) Additionally, there has been no evidence that the MY.NET.97 network itself has been compromised.

```
05/28-19:14:37.565109  [**] SNMP public access [**] MY.NET.97.63:1301 -> MY.NET.101.192:161
05/28-19:15:39.881123  [**] SNMP public access [**] MY.NET.97.63:1302 -> MY.NET.101.192:161
05/28-19:16:42.440903  [**] SNMP public access [**] MY.NET.97.63:1303 -> MY.NET.101.192:161
05/28-19:17:44.802250  [**] SNMP public access [**] MY.NET.97.63:1304 -> MY.NET.101.192:161
05/29-18:27:19.423820  [**] SNMP public access [**] MY.NET.97.183:1051 -> MY.NET.101.192:161
05/29-18:27:21.131280  [**] SNMP public access [**] MY.NET.97.183:1052 -> MY.NET.101.192:161
05/29-18:27:22.201702  [**] SNMP public access [**] MY.NET.97.183:1052 -> MY.NET.101.192:161
05/29-18:27:22.217405  [**] SNMP public access [**] MY.NET.97.183:1053 -> MY.NET.101.192:161
```

4. Tiny Fragments

On several instances, external hosts attempted to send fragmented packets that were smaller than should occur naturally. The following table lists all of the source and destination addresses found in these entries:

Source Address	# of Entries	Destination Address	# of Entries
206.193.209.254	84	MY.NET.219.58	84
24.3.7.221	67	MY.NET.70.121	67
63.236.34.174	6	MY.NET.1.8	6

Attackers will sometimes fragment packets intentionally small in order to pass them through filtering devices that would otherwise drop the packet. If for example, the TCP header is truncated in half, a packet filtering device might pass it on to the host even if it is destined for a disallowed port.

```
05/23-15:24:32.519971 [**] Tiny Fragments - Possible Hostile Activity [**] 206.193.209.254 -> MY.NET.219.58
05/23-15:24:33.012982 [**] Tiny Fragments - Possible Hostile Activity [**] 206.193.209.254 -> MY.NET.219.58
05/23-15:24:38.099890 [**] Tiny Fragments - Possible Hostile Activity [**] 206.193.209.254 -> MY.NET.219.58
05/23-15:24:42.932586 [**] Tiny Fragments - Possible Hostile Activity [**] 206.193.209.254 -> MY.NET.219.58
05/23-15:24:43.245630 [**] Tiny Fragments - Possible Hostile Activity [**] 206.193.209.254 -> MY.NET.219.58
05/23-15:24:44.262475 [**] Tiny Fragments - Possible Hostile Activity [**] 206.193.209.254 -> MY.NET.219.58
05/23-15:24:46.175629 [**] Tiny Fragments - Possible Hostile Activity [**] 206.193.209.254 -> MY.NET.219.58
05/23-15:24:47.995867 [**] Tiny Fragments - Possible Hostile Activity [**] 206.193.209.254 -> MY.NET.219.58
05/23-15:24:48.590209 [**] Tiny Fragments - Possible Hostile Activity [**] 206.193.209.254 -> MY.NET.219.58
```

To mitigate this risk, you should ensure that your packet filtering devices will drop packets that are too small to be naturally occurring.

5. Watchlist Connections

There is a large amount of traffic coming from a foreign network in China and one in Israel. The table details the most prevalent source and destination addresses:

Source Address	# of Entries	Destination Address	# of Entries
159.226.45.3	4743	MY.NET.253.41	4028
212.179.33.224	3146	MY.NET.181.87	3146
159.226.159.1	2954	MY.NET.100.230	2770
212.179.44.36	1782	MY.NET.253.42	1636
212.179.41.10	691	MY.NET.253.43	1179

A considerable amount of this traffic is destined for port 25 (SMTP). This could be indicative of an active attack on the mail servers or of these foreign networks sending hostile e-mail to the MY.NET network.

```
06/01-14:46:40.682610  [**] Watchlist 000222 NET-NCFC [**] 159.226.159.1:2674 -> MY.NET.253.43:25
06/01-14:46:41.917746  [**] Watchlist 000222 NET-NCFC [**] 159.226.159.1:2674 -> MY.NET.253.43:25
06/01-14:46:41.936591  [**] Watchlist 000222 NET-NCFC [**] 159.226.159.1:2674 -> MY.NET.253.43:25
06/01-14:46:43.424081  [**] Watchlist 000222 NET-NCFC [**] 159.226.159.1:2674 -> MY.NET.253.43:25
06/01-14:46:43.427403  [**] Watchlist 000222 NET-NCFC [**] 159.226.159.1:2675 -> MY.NET.253.42:25
06/01-14:46:47.281896  [**] Watchlist 000222 NET-NCFC [**] 159.226.159.1:2675 -> MY.NET.253.42:25
06/01-14:46:48.532084  [**] Watchlist 000222 NET-NCFC [**] 159.226.159.1:2675 -> MY.NET.253.42:25
06/01-14:46:48.543636  [**] Watchlist 000222 NET-NCFC [**] 159.226.159.1:2675 -> MY.NET.253.42:25
06/01-14:46:50.659246  [**] Watchlist 000222 NET-NCFC [**] 159.226.159.1:2675 -> MY.NET.253.42:25
06/01-14:46:52.011940  [**] Watchlist 000222 NET-NCFC [**] 159.226.159.1:2674 -> MY.NET.253.43:25
06/01-14:46:53.463444  [**] Watchlist 000222 NET-NCFC [**] 159.226.159.1:2674 -> MY.NET.253.43:25
```

All traffic to and from these networks should be disallowed given the large volume of traffic coming from them and their specific presence within the snort rules (since they do not appear to have been blocked yet.)

6. NMAP Scan

The compromised host MY.NET.253.12 also appears to be running an NMAP scan as part of its overall scanning effort. It appears that the destination port is a random high numbered port so this is likely an effort at TCP/IP stack fingerprinting to determine the OS of the target (as opposed to enumerating the listening services.)

```
06/01-13:28:49.362157  [**] NMAP TCP ping! [**] MY.NET.253.12:43758 -> MY.NET.102.73:32342
06/01-13:28:53.635992  [**] NMAP TCP ping! [**] MY.NET.253.12:43758 -> MY.NET.102.73:35407
06/01-13:28:56.632772  [**] NMAP TCP ping! [**] MY.NET.253.12:43758 -> MY.NET.102.73:35407
06/01-13:32:18.904953  [**] NMAP TCP ping! [**] MY.NET.253.12:43758 -> MY.NET.102.74:44697
06/01-13:32:23.669483  [**] NMAP TCP ping! [**] MY.NET.253.12:43758 -> MY.NET.102.74:43793
06/01-13:32:26.039215  [**] NMAP TCP ping! [**] MY.NET.253.12:43758 -> MY.NET.102.74:43793
06/01-13:35:37.820606  [**] NMAP TCP ping! [**] MY.NET.253.12:43758 -> MY.NET.102.75:42169
```

The compromise of MY.NET.253.12 should immediately be addressed to prevent any further damage to the network.

7. General Port Scans

The Snort portscan preprocessor is generating a very large number of entries in the log. The portscan preprocessor will create a log entry if a threshold number of ports are being connected to within your network within a given time interval. The threshold number of ports in this system is set to 7 and the time interval is 2 seconds.

```
05/27-02:05:54.185396  [**] spp_portscan: PORTSCAN DETECTED from 202.235.50.12 (THRESHOLD 7 connections
in 2 seconds) [**]
05/27-02:05:55.100141  [**] spp_portscan: portscan status from 202.235.50.12: 56 connections across 56
hosts: TCP(56), UDP(0) [**]
05/27-02:05:55.860890  [**] spp_portscan: portscan status from 202.235.50.12: 55 connections across 55
hosts: TCP(55), UDP(0) [**]
05/27-02:05:56.804220  [**] spp_portscan: portscan status from 202.235.50.12: 53 connections across 53
hosts: TCP(53), UDP(0) [**]
05/27-02:05:57.572766  [**] spp_portscan: portscan status from 202.235.50.12: 57 connections across 57
hosts: TCP(57), UDP(0) [**]
05/27-02:05:58.361731  [**] spp_portscan: portscan status from 202.235.50.12: 50 connections across 50
hosts: TCP(50), UDP(0) [**]
05/27-02:05:59.140873  [**] spp_portscan: portscan status from 202.235.50.12: 53 connections across 53
hosts: TCP(53), UDP(0) [**]
05/27-02:05:59.881040  [**] spp_portscan: portscan status from 202.235.50.12: 62 connections across 62
hosts: TCP(62), UDP(0) [**]
05/27-02:06:00.710415  [**] spp_portscan: portscan status from 202.235.50.12: 55 connections across 55
hosts: TCP(55), UDP(0) [**]
05/27-02:06:01.707889  [**] spp_portscan: portscan status from 202.235.50.12: 47 connections across 47
hosts: TCP(47), UDP(0) [**]
05/27-02:06:02.998446  [**] spp_portscan: portscan status from 202.235.50.12: 39 connections across 39
hosts: TCP(39), UDP(0) [**]
```

The source hosts are conducting these port scans for reconnaissance purposes. A prudent course of action would be to filter against the most common source addresses before they attempt an active attack. Another course of action is to notify the ISP's about the activities of their customers.

Last Updated: October 17th, 2017

Upcoming Training

SANS Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Munich December 2017	Munich, Germany	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
Community SANS Nashville SEC401^	Nashville, TN	Jan 08, 2018 - Jan 13, 2018	Community SANS
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
Las Vegas 2018 - SEC503: Intrusion Detection In-Depth	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	vLive
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS London February 2018	London, United Kingdom	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS Northern VA Spring - Tysons 2018	McLean, VA	Mar 17, 2018 - Mar 24, 2018	Live Event
SANS Secure Canberra 2018	Canberra, Australia	Mar 19, 2018 - Mar 24, 2018	Live Event
SANS 2018	Orlando, FL	Apr 03, 2018 - Apr 10, 2018	Live Event
SANS Baltimore Spring 2018	Baltimore, MD	Apr 21, 2018 - Apr 28, 2018	Live Event
SANS Security West 2018	San Diego, CA	May 11, 2018 - May 18, 2018	Live Event
Community SANS Columbia SEC503	Columbia, MD	Aug 13, 2018 - Aug 18, 2018	Community SANS
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced