Global Information Assurance Certification Paper

GIAC Intrusion Detection Curriculum
Practical Assignment for SANS Security DC 2000

July 5 - 10, 2000
Version 2.2.2

Martin E. Kirwan

TABLE OF CONTENTS

****

Assignment 1- Network Detects (60 Points)

****

Detect 1

```
Sep  6 23:14:55 gordie snort[7401]: IDS278 - SCAN -namedV version probe: 207.46.140.100:4476 -> MY.NET.64.153:53
Sep  7 01:10:34 gordie snort[7508]: IDS278 - SCAN -namedV version probe: 207.46.140.100:9892 -> MY.NET.64.153:53
Sep 13 09:05:49 wilbur snort[4432]: IDS278 - SCAN -namedV version probe: 207.46.140.100:52723 -> MY.NET.64.153:53
```

1. Source of trace

My network.

2. Detect was generated by:

Snort intrusion detection system. The following rule was the trigger:

```
alert udp !$HOME_NET any -> $HOME_NET 53 (msg: "IDS278 - SCAN -namedV version probe"; content:
"|07|version|04|bind|00 0010 0003|";nocase; offset: 12; depth: 32;)
```

This rule triggers when a packet from an external system sends a UDP packet network that contains |07|version|04|bind|00 0010 0003|" to the BIND port on an internal server.

3. Probability the source address was spoofed

The probability of spoofing is low because the sender wants to see the results of his query.

4. Description of attack:

This is a reconnaissance probe to discover the version of BIND that is running on our DNS servers to find an older version that is vulnerable to attacks such as the buffer overflow mentioned on BugTraq. Bind is a popular target because it is the most widely used name server software on the Internet.

5. Attack mechanism:

This is an informational probe. The attacker is querying BIND name servers for the version number. This is usually a precursor to an attack once a vulnerable BIND server is found.

6. Correlations:

Jon Hedlund has reported this traffic previously to GIAC. His post can be found at http://www.sans.org/y2k/090700.htm. The BugTraq Id is 134 and it can be found at http://www.securityfocus.com/frames/?content=/vdb/bottom.html%3Fvid%3D134

7. Evidence of active targeting:

The server being queried is the SOA for out network. So this is a pinpoint query to the most important DNS server on our network.

8. Severity:

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

(5 + 2) – (5 + 1) = 1

Criticality is 5 because this is a core server, the SOA DNS server
Lethality is a 2 because this is only a probe for the version number and not an attack against the daemon.
System Countermeasures gets a 5 because the sever is running the latest OS version and is up to date on patches. The BIND version is also the latest.
I gave Network countermeasures a 1 because even though we do have a restrictive firewall, the firewall will allow this query into the DNS server.

9. Defensive recommendation:

The defenses are fine. It is important to stay on top of the OS patches and keep the BIND daemon up to date.

10. Multiple choice test question, write a question based on the trace and your analysis with your answer.

The intent of this probe is to obtain the BIND version number. Why would a attacker want to know this information?

a) Knowing the named version will allow an attacker to spoof your Domain Name.
b) Knowing the named version will allow an attacker to poison your DNS cache
c) Knowing the named version will allow an attacker to identify the process ID that named is running under and kill it remotely.
d) Knowing the named version will allow an attacker to know exactly whether or not exploit your DNS server is vulnerable to a particular exploit.

Answer: d

Detect 2

secure-0907:Sep 6 23:53:33 gordie snort[7401]: IDS290 - WEB-CGI - infosearch fname: 152.163.213.198:62527 -> MY.NET64.14:80

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
09/06-23:53:33.873549 152.163.213.198:62527 -> MY.NET64.14:80
TCP TTL:52 TOS:0x0 ID:63109
*****PA* Seq: 0x16D0224   Ack: 0x269DBE01   Win: 0x832C
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

1. Source of trace

My Network.

2. Detect was generated by:

The trace was created by a Snort intrusion detection system. The trigger rule was :

alert tcp !$HOME_NET any -> $HOME_NET 80 (msg: "IDS290 - WEB-CGI - infosearch fname"; flags:PA; content: "fname=|";)

This rule looks for any external server to connect to an internal server on port 80 with “name=|”within the packet.

3. Probability the source address was spoofed

The probability of the source address being spoofed is low. The attacker would have wanted to see if the attack was successful.

4. Description of attack:

The attacker is looking for older SGI IRIX servers and attempting to obtain the password file from the server using a documented cgi-bin exploit using a cgi script called `infosrch.cgi.`

5. Attack mechanism:

This cgi script is loaded on SGI IRIX servers and allows the attacker to run commands remotely. Potentially giving a root compromise to the attacker. The infosrch.cgi does not properly user input.

6. Correlations:

This attack is well documented at BugTraq. The exploit is described in Bugtraq here http://www.securityfocus.com/vdb/bottom.html?vid=1031.

7. Evidence of active targeting:

The attacker went after our main web server, however he had not done his homework because the server was not a SGI IRIX server. So the probe did not have a chance. There were no other entries in the access and error logs from this IP Address.

8. Severity:

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

(4 + 1) – (4 + 1) = 0

Criticality is 4 because this is our main web server on the Internet.

Lethality is a 1 because this particular attack is only lethal against SGI IRIX servers. This web server is not.

System Countermeasures gets a 4 because the server is running a fairly recent OS version and should be up to date on patches.

I gave Network countermeasures a 1 because even though we do have a restrictive firewall, the firewall will allow this request to the web server.

9. Defensive recommendation:

Insure that all default cgi-bin files are removed. Verify that the cgi-bin directory is not world writable. Keep the OS and web server up to date and patched.

10. Multiple choice test question, write a question based on the trace and your analysis with your answer.

Many web servers load default cgi-bin programs when the web server is loaded. Removing these programs is important because:

a) The default cgi-bin programs are known to cause viruses

b) The default cgi-bin programs have the incorrect ACL assigned initially.

c) The default cgi-bin programs start when your web server starts.

d) The default cgi-bin programs include well known and documented vulnerabilities.

Answer: d

****

Detect 3

Month	Date	Time	Router IP		AccessList #	Reason	Protocol	Source IP		
Apr	12	6:34:32	MY.NET.64.2 %SEC-6-IPACCESSLOGP:	list	104	denied	tcp	194.217.242.41(1143)	->	MY.NET.64.6(53),
Apr	12	5:52:30	MY.NET.64.2 %SEC-6-IPACCESSLOGP:	list	104	denied	tcp	194.217.242.41(1182)	->	MY.NET.64.6(80),
Apr	12	6:33:44	MY.NET.64.2 %SEC-6-IPACCESSLOGP:	list	104	denied	tcp	194.217.242.41(27950)	->	MY.NET.64.6(25),
Apr	12	6:33:39	MY.NET.64.2 %SEC-6-IPACCESSLOGP:	list	104	denied	tcp	194.217.242.41(3024)	->	MY.NET.64.6(113),
Apr	12	6:36:27	MY.NET.64.2 %SEC-6-IPACCESSLOGP:	list	104	denied	tcp	194.217.242.41(3933)	->	MY.NET.64.6(20),
Apr	12	6:34:09	MY.NET.64.2 %SEC-6-IPACCESSLOGP:	list	104	denied	tcp	194.217.242.41(53)	->	MY.NET.64.6(111),
Apr	12	6:35:27	MY.NET.64.2 %SEC-6-IPACCESSLOGP:	list	104	denied	tcp	194.217.242.41(53)	->	MY.NET.64.6(8080),

1. Source of trace

My network.

2. Detect was generated by:

A border router that is configured to log to a syslog server anytime a deny rule is hit.

3. Probability the source address was spoofed

The probability is low because the scanner would need to see the results to benefit from the scan.

4. Description of attack:

This is a host scan being executed discover what ports are open on the server and what ports are filtered by the border router.

5. Attack mechanism:

The attacker was sending SYN packets to the server to different TCP ports on the single server. The server did not show any connections in its logs so I am presuming that the 3-way handshake was never completed. Once completed this knowledge will enable an attacker to improve his chances of succeeding because he will know what ports are open on the server.

6. Correlations:

I've watched the logs for this IP Address to return but it hasn't yet. No other servers were scanned by this IP address either. The SYN scan method is not very innovative today, but is still widely used.

7. Evidence of active targeting:

The fact that the attacker targeted only this server was disconcerting, especially since I had just moved the server to the Internet the day before. The server was not in the DNS and would have had minimal traffic that it initiated at that time. This was not only active targeting, but also precise targeting.

8. Severity:

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

$(4 + 2) – (5 + 1) = 0$

Criticality is 4 because this was set up to become a mail relay.

Lethality is a 2 because this was only a host scan to discover what ports were accessible from the Internet.

I gave System Countermeasures a 5 because the sever is running the latest OS version and is up to date on patches. The Sendmail version is also the latest.

I gave Network countermeasures a 0 because even though we do have a restrictive firewall, the firewall will allow some of the SYN packets to the server and tell the attacker the ports he can talk on.

9. Defensive recommendation:

Defenses are fine. Keeping on top of OS and Sendmail patches is a must. Added vigilance to the frequency and timing of host scans on new servers we connect to the Internet is a must.

10. Multiple choice test question, write a question based on the trace and your analysis with your answer.

The above trace can best be described as a:

a) TCP scan for well known trojans
b) STEALTH scan for closed ports
c) WIN GATE scan
d) TCP scan for well known services

Answer: d

****

Detect 4

Attacker IP Date Time Command Server response
MY.NET.144.143 - - [25/Aug/2000:14:32:12 -0400] "GET /cgi-bin/phf/?Qalias=x%0a/bin/cat%20/etc/passwd HTTP/1.0" 404 26783 "-"

1. Source of trace

My network.

2. Detect was generated by:

From an access_log on an Apache webserver.

3. Probability the source address was spoofed

The probability is low that the source address was spoofed because the attacker wanted to get the /etc/passwd given to him.

4. Description of attack:

The attacker tried to lift the /etc/passwd file using the /cgi-bin/phf file. This is another default cgi-bin file that gets loaded. If left on the system any user could get any file they could name the directory path to.

5. Attack mechanism:

The attack works when the system administrator has not removed the default phf file from the cgi-bin directory. It allows a remote user to to execute commands.

6. Correlations:

This attack was first announced in 1996 and is documented at http://www.securityfocus.com/frames/?content=/vdb/bottom.html%3Fvid%3D629 . Even though it is so the attack is so old, many scanners and script kiddies still attempt this because system administrators do not properly configure their servers.

7. Evidence of active targeting:

There was active targeting. This server is our main web server on the Internet.

8. Severity:

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

(4 + 1) – (3 + 1) = 2

Criticality is 5 because this is the primary web server for the Internet.

Lethality is a 1 because the phf file had already been removed when the server was configured.

System Countermeasures gets a 3 because the sever is not running the latest OS version but is up to date on patches. The Apache version is also the latest.

I gave Network countermeasures a 1 because even though we do have a restrictive firewall, the firewall will allow this through to the web server.

9. Defensive recommendation:

The OS should be upgrade just to be safe and the logs should continue to be monitored for other attempts.

10. Multiple choice test question, write a question based on the trace and your analysis with your answer.

The phf vulnerability is caused by:

a) An incorrectly compiled web server
b) An incorrectly configured search engine.
c) An incorrectly configured web browser
a) An incorrectly configured server

Answer: d

****

Detect 5

SNORT LOG
Aug 28 15:33:46 wilbur snort[31634]: IDS212 - MISC - DNS Zone Transfer: 199.73.37.2:8782 -> MY.NET.64.4:53
Sep 9 21:19:01 wilbur snort[30159]: IDS212 - MISC - DNS Zone Transfer: 195.188.192.12:4759 -> MY.NET.64.153:53
Sep 9 21:19:16 wilbur snort[30159]: IDS212 - MISC - DNS Zone Transfer: 195.188.192.12:4777 -> MY.NET.64.4:53

DNS SERVER LOG
Log name Action [Attacker IP]:port "zone file requested" (reason)
bind_xferlog:unapproved AXFR from [199.73.37.2].8782 for "prc.com" (acl)
bind_xferlog:unapproved AXFR from [195.188.192.12].2155 for "prc.com" (acl)
bind_xferlog:unapproved AXFR from [195.188.192.12].2677 for "mists.MY.NET" (acl)

1. Source of trace

My network

2. Detect was generated by:

The top set is from a snort IDS box. The bottom three are log entries on the 2 start of authority (SOA) servers. The rule that triggered the detect is

alert tcp !$HOME_NET any -> $HOME_NET 53 (msg:"IDS212 - MISC - DNS Zone Transfer"; content: "|01 00 00 01 00 00 00 00 00 00|"; flags: AP; offset: 2; depth: 16;)

The rule triggers when an external system connects to the BIND port on a server within my network and tries to transfer a zone. The master DNS servers are configured with an ACL that only allows certain slave servers to pull the zone files.

3. Probability the source address was spoofed

Since this is a recon attempt, it is unlikely that the source was spoofed. The attacker would want to receive the results if they had been successful.

4. Description of attack:

This is an attempt to get the zone files for our domain. BIND and DNS servers use the AXFR to transfer the domain zones from a master server to a slave. Allowing an attacker to get the zone files would give him and in-depth knowledge of our network and how and what is deployed throughout it.

5. Attack mechanism:

The attacker is probably using dig or a similar program to get the zone from the two advertised names servers. An example would be

```
dig @ns1.MY.NET axfr MY.NET
```

6. Correlations:

This is a well known recon method and is specifically guarded against in the SANS Securing Linux Step By Step and the O Reilly DNS and BIND book and Cricket Liu's presentation Securing Your Name Server (http://www.acmebw.com/papers/securing.pdf)

7. Evidence of active targeting:

This type of recon has to have active targeting because only the Master or Slave servers will be able to distribute the zone files for a domain.

8. Severity:

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

(5 + 1) – (5 + 1) = 3

Criticality is 5 because these are core servers, the published DNS servers for our domain.

Lethality is a 1 because the DNS servers have ACLs installed that only allowed approved slave DNS servers to pull the zone files.

System Countermeasures gets a 5 because the severs are running the latest OS version and are up to date on patches. The BIND version is also the latest.

I gave Network countermeasures a 1 because even though we do have a restrictive firewall, the firewall will allow this query into the DNS servers.

9. Defensive recommendation:

Defenses are fine, attack was blocked by the ACLs on the DNS servers

10. Multiple choice test question, write a question based on the trace and your analysis with your answer.

AXFR is a method to

a) Reload the zone files on a DNS server
b) Restart the DNS server
c) Update a record in a zone file
d) Transfer a zone file from a DNS server

Answer: d

****

Assignment 2 - Evaluate an Attack (20 Points)

****

1. I have chosen to evaluate the dreaded TELNET command. This is a very insidious reconnaissance tool. It has already infected most of the Internet and most private networks. Many TCP services are susceptible to it and readily give up information such as:
    a. What OS the server is running and version
    b. What daemon is running on a particular port and what version.

2. The attacker can use the telnet client included with his OS or download one of many shareware and freeware clients that are available on the Internet. TELNET can be used against any service that will provide a banner. Examples of susceptible services are telnet, http, smtp. The attacker only needs to pass the IP address or dns name of the victim server and the port of the service that he is interested in to the TELNET command. If the attacker does not pass a port to the TELNET command it assumes that the attacker wishes to connect to the telnet daemon on the victim server. If the service is a available then the server will respond with the appropriate banner. When connecting to the default telnet port the banner usually includes very informative information to an attacker, such as what operating system is running on a server and what version it is. The TELNET client can also be used to get information about services that running on a server such as the daemon name and version number. This information can be used by an attacker to find an exact match between a known vulnerability in a servers OS and the services it provides, and a scripted attack tool.

3. A few examples are :
    a. Typing the command "telnet bert.MY.NET" gives me
        i.
        Hello unknown@kirwan-1.MY.NET
        Red Hat Linux release 6.2 (Zoot)
        Kernel 2.2.16-3 on an i586
        login:

        ii. This banner gives me the operating system name and version. It also gives me the kernel number and the hardware architecture of the server.

    b. Typing the command "telnet bert.MY.NET" gives me:
        i. 220 bert.MY.NET ESMTP Sendmail 8.9.3/8.9.3; Fri, 15 Sep 2000 19:05:38 –0400
        ii. This tells me that sendmail 8.9.3 is running and what the local time and date are and what the variance from GMT is (-400).

c. Typing the command "telnet bert.MY.NET 80" gives me:
   i.

   HTTP/1.1 400 Bad Request
   Date: Fri, 15 Sep 2000 23:10:53 GMT
   Server: Apache/1.3.12 (Unix) (Red Hat/Linux) PHP/3.0.15 mod_perl/1.21
   Connection: close

   ii. This tells me that the server is running Apache version 1.3.12 on a Red Hat Linux box with PHP and mod_perl compiled in. If compared to the sendmail information above you can also get a rough estimate as to where they are geographically by comparing the time/date stamp returned by Sendmail (Fri, 15 Sep 2000 19:05:38 – 0400) and the GMT time returned by the Apache daemon (Fri, 15 Sep 2000 23:10:53 GMT)

****

Assignment 3 - "Analyze This" Scenario (20 Points)

****

To provide a bid to provide security services for this facility. You have been allowed to run a Snort system with a fairly standard rulebase for a month. From time to time, the power has failed, or the disk was full so you do not have data for all days. Your task is to analyze the data, be especially alert for signs of compromised systems or network problems and produce an analysis report.

****

To Whom It May Concern,

Thank you for allowing my company to provide a sample of our security services. The information below is preliminary without knowledge of the network architecture and the normal traffic patterns. I have taken some of the alerts that appeared in the logs and given my recommendations for further actions. Please feel free to contact me if you have any further questions.

Thank You,

Marty Kirwan

****

Alert 1 - Portscan: The possibility exists that MY.NET253.12 has been compromised. It has triggered the portscan settings within the snort device. The triggers that it tripped were the use of NMAP and the number of connects. This device was involved with over 97,000 entries in the log files.

```
05/28-16:50:46.509072  [**] NMAP TCP ping! [**] MY.NET.253.12:43758 -> MY.NET.16.41:39982
05/28-17:04:33.623267  [**] spp_portscan: portscan status from MY.NET.253.12: 2 connections across 1 hosts: TCP(2), UDP(0) ST
EALTH [**]
05/28-16:50:49.483886  [**] NMAP TCP ping! [**] MY.NET.253.12:43758 -> MY.NET.16.41:39982
05/28-17:04:35.161858  [**] spp_portscan: portscan status from MY.NET.253.12: 3 connections across 1 hosts: TCP(2), UDP(1) ST
EALTH [**]
05/28-17:02:53.899558  [**] spp_portscan: portscan status from MY.NET.253.12: 27 connections across 1 hosts: TCP(27), UDP(0)
[**]
05/28-17:02:55.869332  [**] spp_portscan: portscan status from MY.NET.253.12: 33 connections across 1 hosts: TCP(33), UDP(0)
[**]
05/28-17:02:57.298090  [**] spp_portscan: portscan status from MY.NET.253.12: 42 connections across 1 hosts: TCP(42), UDP(0)
[**]
05/28-17:02:59.187248  [**] spp_portscan: portscan status from MY.NET.253.12: 33 connections across 1 hosts: TCP(33), UDP(0)
[**]
05/28-17:03:01.277051  [**] spp_portscan: portscan status from MY.NET.253.12: 46 connections across 1 hosts: TCP(46), UDP(0)
```

The possibility exists that this box is being used be an internal security group and that this traffic is authorized. It is my recommendation that the owners of this box be identified and that is be examined for compromise.

Alert 2 - SYN-FIN scan: Another high concern I have with the traffic in the logs is that amount of scanning that is being flagged from without the network to the internal. It is assumed that the snort device has been placed within the network perimeter because of the traffic between different MY.NET networks. Too much non-targeted traffic is making its way to the internal network that the snort device is located on. Examples include:

06/13-01:30:39.951276 [**] SYN-FIN scan! [**] 204.60.176.2:53 -> MY.NET.1.9:53
06/13-01:30:39.964928 [**] SYN-FIN scan! [**] 204.60.176.2:53 -> MY.NET.1.10:53
06/13-01:30:39.991306 [**] SYN-FIN scan! [**] 204.60.176.2:53 -> MY.NET.1.11:53

This trace appears to be a scan for a dns server within the MY.NET network. The scan is not targeted at any particular server. It is using a target port of 53 to get through filtering devices. It also has the SYN and FIN flags set to avoid completion of the three-way handshake to avoid logging by the host. The scanner might also be looking for rogue dns servers that were installed with an OS system and never turned off. This particular scanner had scanned for over 13000 hosts within the time period of the logs. This is out of the 18260 total SYN-FIN scans conducted.

I have two recommendations concerning this alert. I recommended that an internal scan for these rogue dns servers be performed to identify them and turn them off. More importantly, it is recommended that a firewall or filtering router be placed at the network perimeter to only allow traffic to the approved services sitting on approved servers.

Alert 3 - SMB Name Wildcard : A related issue to the second alert is the amount of SMB Name Wildcard traffic that is being sent from the Internet to the internal network. This traffic could be legitimate attempts by Windows devices to find a hosts name. However, this traffic should be knocked down at the perimeter by a filtering device because it can be used as a reconnaissance method to map out your network and identify Windows devices.

05/24-20:53:43.074063 [**] SMB Name Wildcard [**] 166.90.30.149:137 -> MY.NET.100.130:137
05/24-20:53:46.531240 [**] SMB Name Wildcard [**] 166.90.30.149:137 -> MY.NET.100.130:137
05/25-13:32:24.210597 [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137
05/25-13:32:24.226563 [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137
05/28-18:45:06.630606 [**] SMB Name Wildcard [**] MY.NET.101.89:137 -> MY.NET.70.234:137
05/28-18:45:54.040922 [**] SMB Name Wildcard [**] MY.NET.70.234:137 -> MY.NET.101.145:137
05/28-18:45:54.041819 [**] SMB Name Wildcard [**] MY.NET.101.145:137 -> MY.NET.70.234:137
05/28-18:46:15.377066 [**] SMB Name Wildcard [**] MY.NET.70.234:137 -> MY.NET.101.147:137
05/28-18:46:15.382603 [**] SMB Name Wildcard [**] MY.NET.101.147:137 -> MY.NET.70.234:137

Alert 4 - Watchlist 000222 NET-NCFC: One of the alerts that the snort device was configured to watch was a Watchlist group. This includes a lot of traffic from mail servers within China to what appears to be internal mail servers win MY.NET.

```
05/22-08:25:19.757032  [**] Watchlist 000222 NET-NCFC [**] 159.226.45.3:25 -> MY.NET.6.7:17207
05/22-08:25:20.693372  [**] Watchlist 000222 NET-NCFC [**] 159.226.45.3:25 -> MY.NET.6.7:17207
05/22-08:25:32.577902  [**] Watchlist 000222 NET-NCFC [**] 159.226.45.3:25 -> MY.NET.6.7:17207
06/22-18:12:51.945488  [**] Watchlist 000222 NET-NCFC [**] 159.226.5.222:1447 -> MY.NET.100.230:113
06/22-18:12:58.137481  [**] Watchlist 000222 NET-NCFC [**] 159.226.5.222:1462 -> MY.NET.100.230:113
06/22-18:13:14.394802  [**] Watchlist 000222 NET-NCFC [**] 159.226.5.222:1500 -> MY.NET.100.230:113
06/23-10:18:19.646860  [**] Watchlist 000222 NET-NCFC [**] 159.226.45.3:2084 -> MY.NET.253.42:25
06/23-10:18:19.657890  [**] Watchlist 000222 NET-NCFC [**] 159.226.45.3:2084 -> MY.NET.253.42:25
06/23-10:18:19.663125  [**] Watchlist 000222 NET-NCFC [**] 159.226.45.3:2084 -> MY.NET.253.42:25
```

The creation of a rule set to trigger upon any traffic to or from the 159.226.0.0 domain implies that you already have some suspicion with the traffic go to this domain. It appears that the hosts within the MY.NET network are allowed to send and receive their own mail directly to and from the Internet. It is my recommendation that a set of mail hubs and mail relays be set up to handle the SMTP traffic to and from the Internet for the MY.NET network and turn down SMTP access to and from the Internet for all other hosts. The IP addresses within the logs files should be investigated to ensure that they have not been compromised.

Alert 5 - Tiny Fragments: The tiny fragments rule was triggered by 3 servers. Tiny fragments could be an indication of covert channels within your network. It is also possible that there is or was a network issue that caused this traffic to fragment. It is impossible to determine this without seeing the actual packets. It is my recommendation that these external and internal IP addresses be placed onto the Watchlist for further investigation. It is also recommended that all tiny fragments be recorded for reference.

```
05/23-15:26:12.814314  [**] Tiny Fragments - Possible Hostile Activity [**] 206.193.209.254 -> MY.NET.219.58
05/23-15:26:13.194403  [**] Tiny Fragments - Possible Hostile Activity [**] 206.193.209.254 -> MY.NET.219.58
05/23-15:26:13.332578  [**] Tiny Fragments - Possible Hostile Activity [**] 206.193.209.254 -> MY.NET.219.58
05/23-15:26:17.061704  [**] Tiny Fragments - Possible Hostile Activity [**] 206.193.209.254 -> MY.NET.219.58
05/27-15:16:59.885167  [**] Tiny Fragments - Possible Hostile Activity [**] 24.3.7.221 -> MY.NET.70.121
05/27-15:17:00.724828  [**] Tiny Fragments - Possible Hostile Activity [**] 24.3.7.221 -> MY.NET.70.121
05/27-15:17:01.137635  [**] Tiny Fragments - Possible Hostile Activity [**] 24.3.7.221 -> MY.NET.70.121
05/27-15:17:01.855067  [**] Tiny Fragments - Possible Hostile Activity [**] 24.3.7.221 -> MY.NET.70.121
```

Alert 6 - WinGate 8080 Attempt: It appears that many hosts are scanning the MY.NET network looking for open proxy servers.

```
06/01-02:13:16.954766  [**] WinGate 8080 Attempt [**] 202.38.128.188:3785 -> MY.NET.105.32:8080
06/01-02:13:16.955046  [**] WinGate 8080 Attempt [**] 202.38.128.188:3787 -> MY.NET.105.34:8080
06/01-02:13:17.030959  [**] WinGate 8080 Attempt [**] 202.38.128.188:3817 -> MY.NET.105.64:8080
06/01-02:13:17.034979  [**] WinGate 8080 Attempt [**] 202.38.128.188:3818 -> MY.NET.105.65:8080
06/01-02:13:17.050758  [**] WinGate 8080 Attempt [**] 202.38.128.188:3819 -> MY.NET.105.66:8080
06/01-02:13:17.056826  [**] WinGate 8080 Attempt [**] 202.38.128.188:3820 -> MY.NET.105.67:8080
06/01-02:13:17.076534  [**] WinGate 8080 Attempt [**] 202.38.128.188:3821 -> MY.NET.105.68:8080
06/01-02:13:17.091683  [**] WinGate 8080 Attempt [**] 202.38.128.188:3822 -> MY.NET.105.69:8080
06/01-02:13:17.095652  [**] WinGate 8080 Attempt [**] 202.38.128.188:3794 -> MY.NET.105.41:8080
06/01-02:13:17.207716  [**] WinGate 8080 Attempt [**] 202.38.128.188:3823 -> MY.NET.105.70:8080
06/01-02:13:17.216724  [**] WinGate 8080 Attempt [**] 202.38.128.188:3824 -> MY.NET.105.71:8080
06/01-02:13:17.280596  [**] WinGate 8080 Attempt [**] 202.38.128.188:3825 -> MY.NET.105.72:8080
06/01-02:13:17.311094  [**] WinGate 8080 Attempt [**] 202.38.128.188:3826 -> MY.NET.105.73:8080
06/01-02:13:17.405292  [**] WinGate 8080 Attempt [**] 202.38.128.188:3827 -> MY.NET.105.74:8080
06/01-02:13:17.410755  [**] WinGate 8080 Attempt [**] 202.38.128.188:3828 -> MY.NET.105.75:8080
```

This trace is an example. 202.38.128.188 scanned over 22000 hosts within the MY.NET network just within the time covered by the logs. This IP has been assigned to the Chinese Academy of Sciences and is further evidence of suspicious traffic to and from China. This is another example of traffic that could be prevented from entering the MY.NET network by a filtering device at the perimeter.

Alert 7 - SNMP public access: It appears that MY.NET.101.192 is setup with the default SNMP settings. These setting should be changed to something besides public and private to keep the curious from accessing the device using SNMP.

Alert 8 – GIAC 08-feb-2000: This alert appears to have been set up to trigger upon any traffic destined for MY.NET.179.77. This may be watched because it has previously had suspicious activity or is a honey pot sitting out there waiting for someone to try and connect. Without knowing the reason for the creation of this rule I can only recommend further surveillance.

```
05/28-06:20:44.976398  [**] GIAC 08-feb-2000 [**] 195.11.50.204:2125 -> MY.NET.179.77:554
05/28-06:21:07.149318  [**] GIAC 08-feb-2000 [**] 195.11.50.204:53 -> MY.NET.179.77:53
05/28-06:21:14.732025  [**] GIAC 08-feb-2000 [**] 195.11.50.204:1097 -> MY.NET.179.77:119
```

Alert 9 - GIAC 000218 VA-CIRT port [34555 and 35555]: This alert appears to be set up to trigger on any traffic destined for ports 34555 or 35555. This may have been set up to search for a suspected Trojan. Without knowing the reason for the creation of this rule I can only recommend further surveillance

05/24-21:36:12.657095 [**] GIAC 000218 VA-CIRT port 35555 [**] 208.210.124.27:25 -> MY.NET.253.41:35555
05/24-21:36:12.759486 [**] GIAC 000218 VA-CIRT port 35555 [**] 208.210.124.27:25 -> MY.NET.253.41:35555
05/25-01:47:17.966506 [**] GIAC 000218 VA-CIRT port 34555 [**] 209.38.76.60:113 -> MY.NET.6.34:34555
05/25-01:47:18.100379 [**] GIAC 000218 VA-CIRT port 34555 [**] 209.38.76.60:113 -> MY.NET.6.34:34555

Last Updated: May 27th, 2017

Upcoming Training

Security Operations Center Summit & Training	**Washington, DC**	**Jun 05, 2017 - Jun 12, 2017**	**Live Event**
SANS Houston 2017	**Houston, TX**	**Jun 05, 2017 - Jun 10, 2017**	**Live Event**
SANS Columbia, MD 2017	**Columbia, MD**	**Jun 26, 2017 - Jul 01, 2017**	**Live Event**
SANS London July 2017	**London, United Kingdom**	**Jul 03, 2017 - Jul 08, 2017**	**Live Event**
SANSFIRE 2017	**Washington, DC**	**Jul 22, 2017 - Jul 29, 2017**	**Live Event**
SANSFIRE 2017 - SEC503: Intrusion Detection In-Depth	**Washington, DC**	**Jul 24, 2017 - Jul 29, 2017**	**vLive**
SANS San Antonio 2017	**San Antonio, TX**	**Aug 06, 2017 - Aug 11, 2017**	**Live Event**
SANS Boston 2017	**Boston, MA**	**Aug 07, 2017 - Aug 12, 2017**	**Live Event**
SANS Adelaide 2017	**Adelaide, Australia**	**Aug 21, 2017 - Aug 26, 2017**	**Live Event**
SANS Virginia Beach 2017	**Virginia Beach, VA**	**Aug 21, 2017 - Sep 01, 2017**	**Live Event**
SANS Network Security 2017	**Las Vegas, NV**	**Sep 10, 2017 - Sep 17, 2017**	**Live Event**
SANS vLive - SEC503: Intrusion Detection In-Depth	**SEC503 - 201709,**	**Sep 11, 2017 - Oct 18, 2017**	**vLive**
Baltimore September 2017 - SEC503: Intrusion Detection In-Depth	**Baltimore, MD**	**Sep 25, 2017 - Sep 30, 2017**	**vLive**
SANS Baltimore Fall 2017	**Baltimore, MD**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
SANS London September 2017	**London, United Kingdom**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
Community SANS Scottsdale SEC503	**Scottsdale, AZ**	**Oct 02, 2017 - Oct 07, 2017**	**Community SANS**
SANS October Singapore 2017	**Singapore, Singapore**	**Oct 09, 2017 - Oct 28, 2017**	**Live Event**
Community SANS Boston SEC503	**Boston, MA**	**Oct 09, 2017 - Oct 14, 2017**	**Community SANS**
Community SANS Ottawa SEC503	**Ottawa, ON**	**Oct 16, 2017 - Oct 21, 2017**	**Community SANS**
SANS Seattle 2017	**Seattle, WA**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
SANS San Diego 2017	**San Diego, CA**	**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**
SIEM & Tactical Analytics Summit & Training	**Scottsdale, AZ**	**Nov 28, 2017 - Dec 05, 2017**	**Live Event**
SANS Cyber Defense Initiative 2017	**Washington, DC**	**Dec 12, 2017 - Dec 19, 2017**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**