Global Information Assurance Certification Paper

GIAC INTRUSION DETECTION IN DEPTH

GCIA Practical Assignment v3.2

by Ewen Fung

December 15, 2002

TABLE OF CONTENT

ASSIGNMENT 1 – HONEYPOT DESIGN AND PRACTICAL SETUP GUIDE USING HONEYD

Abstract

This paper discusses the design and implementation considerations of honeypot, with more focus on the concerns of obviousness of the honeypot. In order to address this concern, the importance of TCP stack emulation feature provided by honeypot is highlighted. With the demonstration tool Honeyd, the only one Open Source honeypot that deals with the TCP stack emulation up to the moment writing this paper, the importance of TCP stack emulation has been shown. However, this paper is not intended to discuss the overall usefulness of a honeypot as an intrusion detection tool.

Practical installation steps for Honeyd on RedHat Linux v7.3 has been shown in this paper. Results of port scanning, OS detection and vulnerability testing on Honeyd sample configuration and fine-tuned configuration have also been shown in this paper, in order to demonstrate the configuration of Honeyd and how to configure the TCP stack emulation feature.

Introduction

Honeypot is a variant of standard Intruder Detection Systems (IDS) but with more focus on information gathering and deception. When the intruder is decoying by the honeypot, the honeypot can be used to log their activities for analyzing their techniques and to gather more information for further investigation.

However, building a honeypot is not as simply as installing an "out-of-box" operating systems and applications (i.e. new system without patches installed and/or without changing the default configurations). Considerations on its self-protection, logging capability, attack detection, alert mechanism and most importantly the emulation of services and OS are some of the important factors on the success of the honeypot.

Without consider the obviousness of the honeypot, this may raise suspicions and intruders may be not trapped finally. One of the most common "mistakes" is that emulating a vulnerable NT service in a Linux box. The reason why I say this is a "mistake" will be demonstrated later in this paper.

What is honeypot?

This paper does not intend to re-state the technology of honeypot, but for completeness of this paper, I would like to state my definition of honeypot here. With the solid definition of Internet Trap by Edward Amoroso [5], I would like to make some modification and define Honeypot as follows:

A honeypot is a deception-based intrusion detection tools that targets to

- *emulate an asset that contains valuable information; and/or*

- *to divert the activities of a potential intruder from the real, valued assets to bogus assets,*

for the purposes of:

- *decoying or deterring the intruder;*
- *gathering intrusion-related information;*
- *delaying the intrusion activities; and/or*
- *initiating necessary response.*

In order to achieve the purposes of a honeypot, the most critical factor is to attract the intruder and lie that the 'valuable asset' was readily available by a simple exploitation or other techniques. Also, it should also lie that it was successfully attacked or compromised.

Without accesses by intruders, the honeypot is worthless. Also, the honeypot is not a kind of safeguard for your organization and it cannot protect your systems. [5] These concepts should be aligned while you read this paper.

Design and implementation considerations of honeypot

In the design and implementation of a honeypot, I personally feel that certain areas must be considered and well prepared before putting the honeypot in the "live" environment.

Design Considerations

Deploy with current best practices in your organization

Scanning tools are a common means that the intruders used for network reconnaissance. Intruders usually use these kinds of tools to map the target network and gather more necessary information, such as IPs of hosts alive, before the actual attacks.

For this reason, it is recommended that the honeypot should be configured and protected according to the your existing security policy. e.g. protect the honeypot by a firewall with only several "useful" ports accessible by the "public".

Mirror the real asset as much as possible

As mentioned, intruders may do some researches on the target network before the actual attacks. It is obvious that the honeypot should be setup exactly, or nearly, the same as a real asset. Suspicion may be raised if the "asset" is readily accessible to the public without any safeguard in place. There are several recommendations for this issue:

1. Configure the honeypot with TCP stack emulation for reasonableness

   Since different OS may have different implementation on the exceptions of TCP/IP packets. By looking for things that differ among OS, writing a probe for the difference and combining enough of these, you can narrow down the OS very

tightly. With such OS fingerprinting techniques, it is possible to detect the remote host's OS even it has limited TCP or UDP ports opened.

There are quite a lot of Open Source and commercial honeypot available. Some of them are emulating an out-of-box system with full functionality, while some of them are emulating the vulnerable services, such as unpatched IIS web server, using shell scripts or interpreter. The latter kind of honeypot usually focus on the emulation of vulnerable services and lack the feature of OS emulation, i.e. TCP stack emulation.

Without altering the TCP stack responses, experience hackers may not be decoyed due to the uncertainties or conflicting reconnaissance results. An example of this will be a result of vulnerable MS IIS web server running on a RedHat Linux 6.2. Anyway, you can say that this can keep the hacker away from attacking your network but this should not be the main purpose of our honeypot.

Honeyd is an Open Source honeypot, which introduces a variety of new concepts including the ability to monitor millions of unused IPs, IP stack spoofing, and simulate hundreds of operating systems, at the same time. The setup and demonstration will be shown in the coming section.

2. Ensure the consistence of the setting and design of the honeypot.

   The configuration and design of the honeypot should be consistent. If you are emulating a web server of your company, you may consider putting part of your company web pages in the honeypot. Similarly, if you are emulating a FTP server for technical support purpose, you may put some related technical documents there.

   If the IP address was used as a "vulnerable web server", be sure not to immediately reconfigure the honeypot as a "vulnerable FTP server" as it may raise the suspicion to the intruders. You may consider using other IP when reconfigure the honeypot while putting the old honeypot as transition.

Reconfigure for capturing different information

It is also recommended to reconfigure the honeypot periodically so that you can capture different hacking techniques, understand the interests from the intruders point of views and most importantly remove the suspicion from the intruders.

For further explanation on the consistence of the honeypot, it is also recommended that you may reconfigure the current honeypot to emulate the same asset with "patches" installed so that it is not vulnerable anymore – pretending the "asset" is under administration. Another honeypot with other "vulnerable" service will then be enabled for starting the other deception.

Technical Consideration

Self-protection Mechanism

It is very important to configure the honeypot properly in order to prevent it to be the platform for the intruders to launch another attacks. This not only exposes another risk to your servers and network but may also affect your organization goodwill if attacked was identified to be launching from your network.

It is recommended that the honeypot should run in a jailed environment so that the real environment can be separated even though the honeypot itself really compromised.

Remote logging capability

As the honeypot contains valuable data such as the intrusion activity event logs, it is recommended that the logging can be sent to a remote log server rather than 'stdout' or local log file.

Once the honeypot itself was really compromised, intruder may have the ability to remove the log files, which is one of the steps to hide their activities. With remote logging facility such as syslog service, all the important data can be stored in a remote, secured machine for further investigation.

Alert mechanism

I personally think that an inspect function, with instant alert, on the honeypot activities is very important. Whenever the inspect function detected certain user-defined patterns or well-known attacks occurred, the honeypot should immediately notify the security administrator via mail or SMS for further investigation. With instant notification, security administrator can perform some pre-defined tasks such as trace back the source address of the intruder or closely monitor the activities of the intruder.

Keep them online for extended period

The use of deception mechanisms to keep attackers online for extended periods of time is another way to assist in tracing back and gathering more information about their activities. "Well organized" directory structure that contains the organization's financial statement may be used to keep the intruder searching the "data".

Introduction of HoneyD

After the understanding of the overall honeypot design and technical considerations, a lab with setup guide on HoneyD will be shown in the coming section. The setup mainly focuses on the important feature of the HoneyD – TCP stack emulation. The reason for selecting HoneyD is that it is the only one Open Source honeypot that deals with the TCP stack emulation up to the moment writing this paper. Other than that, HoneyD can emulate more than thousand of virtual hosts by using a single machine at the same time. Each virtual host can be configured, via simple

configuration file, so that it listens to arbitrary services accordingly. For the details of the feature, please refer to the section below, or HoneyD official web page.

Practical setup guide of HoneyD on RedHat Linux 7.3

Download the packages needed

HoneyD
Download the honeyd-0.3.tar.gz (Release 2002-07-30)
http://www.citi.umich.edu/u/provos/honeyd/honeyd-0.3.tar.gz

ARPD
Download the arpd-0.1.tar.gz (Release 2002-04-15)
http://www.citi.umich.edu/u/provos/honeyd/arpd-0.1.tar.gz

Library Dependencies
Download the libevent-0.5.tar.gz (Release 2002-06-12)
http://www.monkey.org/~provos/libevent-0.5.tar.gz

Download the libdnet-1.4.tar.gz (Release 2002-04-15)
http://prdownloads.sourceforge.net/libdnet/libdnet-1.4.tar.gz
OR http://libdnet.sourceforge.net and follow the download link

Honeyd Basic Installation

1. While downloading the packages above, save all files under the same directory, e.g. `/honeyd_packages`

2. Substitute with user 'root' if you are currently not.
   ```
   # /bin/su -
   ```
   and provide the root password.

3. Change to the directory `/honeyd_packages`
   ```
   # cd /honeyd_packages
   ```

4. Extract the packages **libdnet**:
   ```
   # tar -zvxf libdnet-1.4.tar.gz
   ```

5. Compile the **libdnet**:
   ```
   # cd libdnet-1.4      (Note: pwd is /honeyd_packages/ libdnet-1.4)
   # ./configure
   # make
   # make install        (Note: Quick verification by checking if 'libdnet.a' is
                          installed in /usr/local/lib/ after 'make install')
   ```

6. Extract the packages **libevent**:
   ```
   # cd ..
   # tar -zvxf libevent-0.5.tar.gz
   ```

7. Compile the **libevent**:

```
# cd libevent          (Note: pwd is /honeyd_packages/libevent)
# ./configure
# make
# make install         (Note: Quick verification by checking if 'libevent.a' is
                        installed in /usr/local/lib/ after 'make install')
```

8. Extract the packages **arpd**:

```
# cd ..
# tar -zvxf arpd-0.1.tar.gz
```

9. Compile the **arpd**:

```
# cd arpd              (Note: pwd is /honeyd_packages/arpd)
# ./configure
# make
# make install         (Note: Quick verification by checking if 'arpd' is
                        installed in /usr/local/sbin after 'make install')
```

10. Extract the packages **Honeyd** to `/honeyd`:

```
# mv /honeyd_packages/honeyd-0.3.tar.gz /
# cd /                 (Note: pwd is /)
# tar -zvxf honeyd-0.3.tar.gz
```

11. Compile the **Honeyd**:

```
# cd honeyd            (Note: pwd is /honeyd)
# ./configure
# make                 (Note: We will not run 'make install' so that all
                        Honeyd binary and scripts will be placed in /honeyd)
# chown -R root:root /honeyd
```

12. Remove the packages:

```
# rm -rf /honeyd_packages
```

Up to this step, the Honeyd should be able to start with sample configuration.

13. To verify the Honeyd installation with sample configuration:

```
# /usr/local/sbin/arpd <ip.addr.your_net/netmask>
```

(e.g. `# arpd -i eth1 10.x.y.253/32`)

```
arpd[19057]: listening on eth1: arp and dst net
10.x.y.253/32 and not ether src xx:xx:xx:xx:xx:xx
```

This command tells your Ethernet card (`eth0` in this example) to response to all traffic send to the IP/Network address (10.x.y.253 in this example) specified, and which the MAC address is not the MAC address of your Ethernet card. Be careful to use an IP address that is not in used within the subnet.

```
# vi config.sample     (Note: pwd is /honeyd)
```

Locate the line that read '`bind 10.1.1.1 template`'. Replace the IP address with 10.x.y.253 and save the file. The parameters of the configuration file will be explained later in this paper.

```
# honeyd -d -p ./nmap.prints -f config.sample \
10.x.y.253/32
```

```
honeyd[19068]: listening on eth1: (tcp or icmp or
udp) and dst net 10.x.y.253/32 and not ether src
xx:xx:xx:xx:xx:xx
```

This command tells the Honeyd to start as NOT a daemon with verbose debug output on screen (-d), use the fingerprint file nmap,prints (-p ./nmap.prints), with the sample configuration file config.sample (-f config.sample) and listen to the IP address 10.x.y.253/32.

14. After started the Honeyd, you can try to connect to the port 80 of 10.x.y.253 from the machine in the 10.x.y.0/24 network segment and check the emulation of web server by the Honeyd. Figure 1 below shows the response from the default web daemon script.

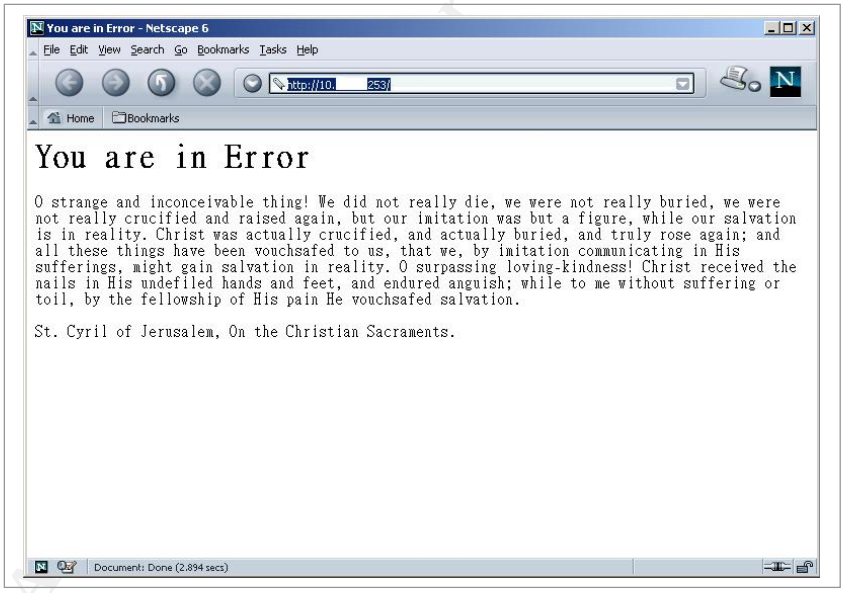


Fig.1 Web server emulation using default Honeyd script

OS detection result on sample configuration

After the setup of the Honeyd using the sample configuration, Nmap v.3.0.0 and ISS Internet Scanner v.6.2.1 have been used to scan that honeypot in order to check what information will be obtained.

Nmap detected that the remote host (the honeypot) is running AIX 4.0 – 4.2 by using the sophisticated OS fingerprinting technique. When you take a look on the sample

configuration of the Honeyd, the line '`annotate "AIX 4.0 - 4.2" fragment old`' tells the Honeyd to emulate the TCP stack as AIX 4.0 – 4.2. Besides, the Nmap fingerprint file is used by the Honeyd in order to emulate the TCP stack, it is obvious that Nmap detected that the honeypot is running AIX 4.0 – 4.2. Figure 2 below shows the result of OS detection using Nmap.

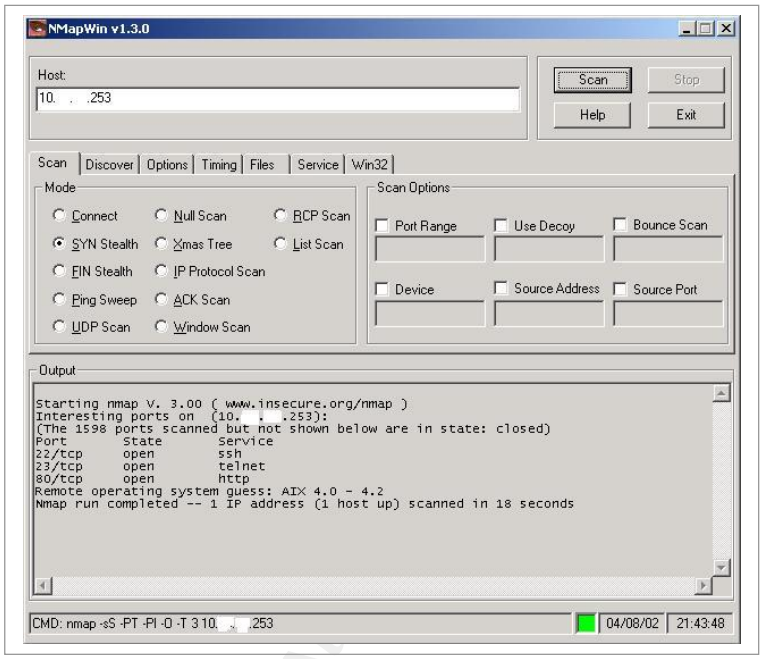


Fig.2 OS detection result using Nmap v.3.0.0

It seems that the honeypot is setup as expected. Next target is to check if the honeypot is emulating a vulnerable web server. ISS Internet Scanner has been used and it detected that the remote host is running UNIX. However it cannot detects the exact OS of the honeypot. Anyway, the honeypot is emulating an UNIX host with a vulnerable web server.

Although the vulnerable web server was found, the vulnerability found was related to "Microsoft IIS Unicode Translation". Can Microsoft IIS run on AIX UNIX? Obviously it wouldn't be the fact. Figure 3 below shows the result from the ISS Internet Scanner.

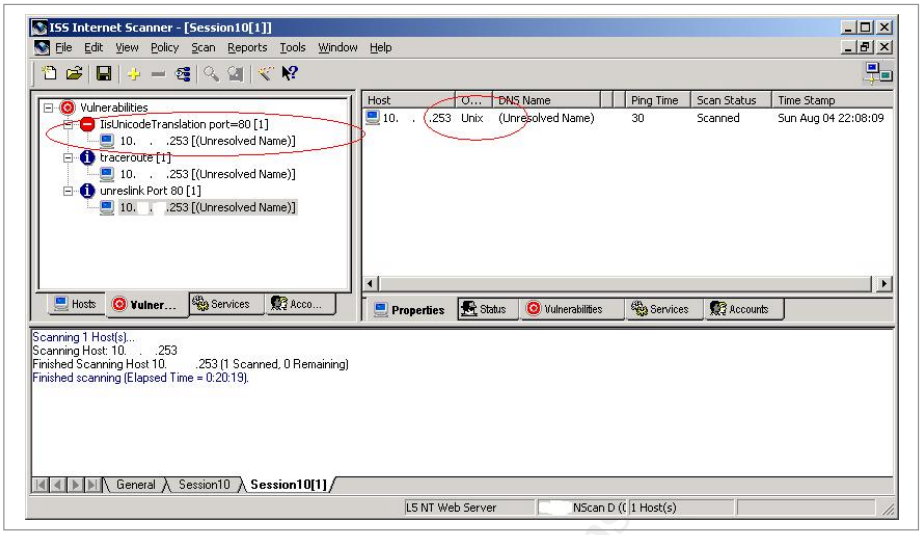


Fig 3. ISS Internet Scanner Result – IIS Vulnerability on UNIX host

Fine tuning on Honeyd configuration

For demonstration on the TCP stack emulation, the fine-tuning processes try to setup a "vulnerable" IIS web server running Windows 2000 server.

Download the IIS emulation script that is contributed by Rain Forest Puppy & HDM. http://sourceforge.net/projects/iisemul8/ (Release 2002-07-20) and save it in `/honeyd_packages` directory.

1. Extract and install the **iisemul8**:

```
# cp /honeyd_packages/iisemulator-0.95.tar.gz /
# cd /
# tar -zvxf iisemulator-0.95.tar.gz
# cd iisemulator-0.95
# cp /honeyd/honeyd /iisemulator-0.95
# chown -R root:root /iisemulator-0.95
# cd /iisemulator-0.95
```

2. Edit the `docs/honeyd.conf` to setup the "Windows 2000" and "IIS web server" so that the final configuration looks like this:

```
# Create a Profile of the virtual server
create profile_name
# Set this virtual server as MS Windows2000
# Refer to nmap.prints for the personality name
set profile_name personality "MS Windows2000 Professional
\ RC1/W2K Advance Server Beta3"
# Add a "service" TCP/80 to the virtual server
# when connected, run the script iisemul8.pl with PERL
add profile_name tcp port 80 "/usr/bin/perl iisemul8.pl"
```

```
# Add a "service" TCP/25 to the virtual server
# when connected, run the script smtp.sh which emulate
# a Sendmail SMTP Server
add template tcp port 25 "/bin/sh \
/honeyd/scripts/smtp.sh"
# Add a "service" TCP/110 to the virtual server
# when connected, run the script which emulate a POP3 Srv
add template tcp port 110 "/bin/sh \
/honeyd/scripts/pop3.sh"
# Set the default action for TCP request to RESET
set profile_name default tcp action reset
set profile_name uid 32767 gid 32767
# Bind this virtual server to 10.x.y.253
bind 10.x.y.253 profile_name
set 10.x.y.253 uptime 1327650
```

OS detection result on fine tuned configuration

With the TCP stack emulation configuration, we have tested the honeyd with Nmap again. Figure 4 below shows the port scanning and OS detection result using Nmap.

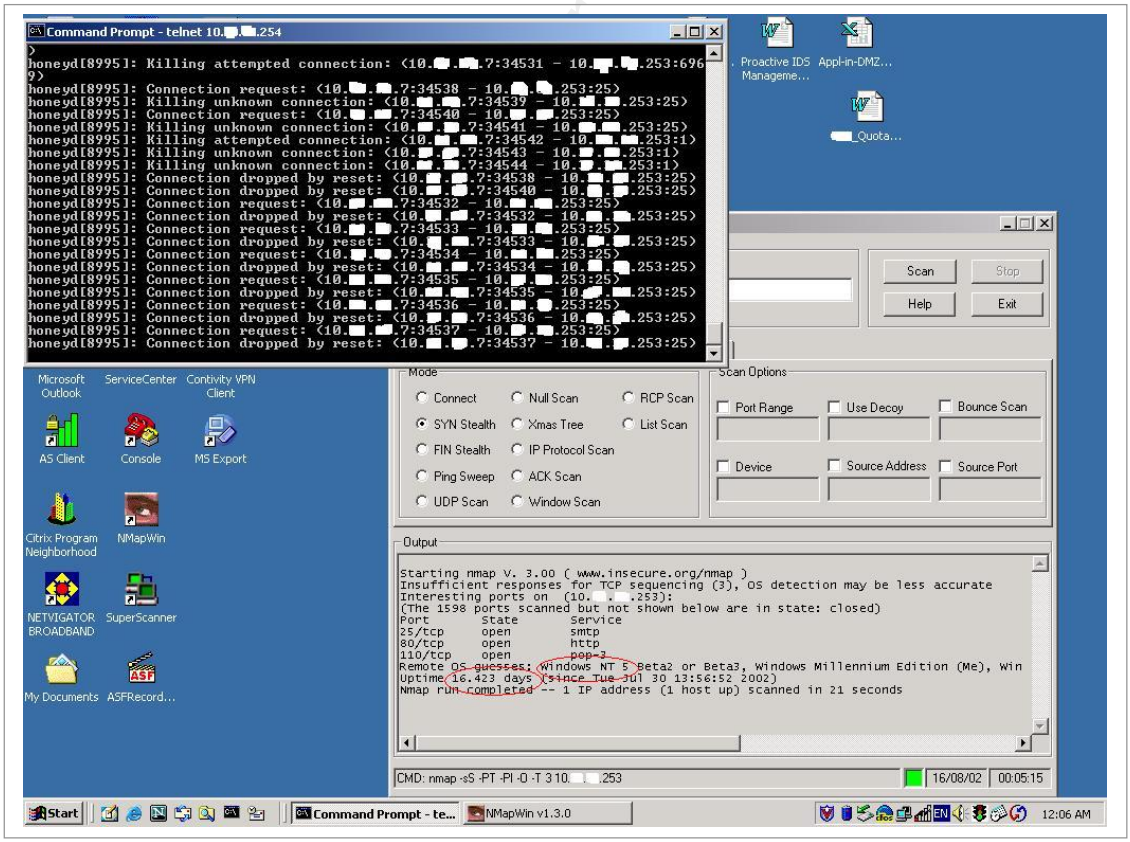


Fig.4 OS detection result using Nmap v.3.0.0 with TCP stack emulation enabled

It is shown that the RedHat Linux 7.3 has successfully run the honeyd, which configured as a IIS web server, SMTP server and POP3 server. (Note that for

demonstration reason, I have enabled the “SMTP” and “POP3” services, which seems to be services running on UNIX host. However, we only focus on the OS detection result.) The Nmap guessed that the remote OS is Windows NT 5 Beta 2 or 3 or Windows ME instead of AIX now. We have successfully implemented the installation of an “IIS Web Server” running on “Windows 2000”!

Opinions to Honeyd

Honeyd obviously provides lots of useful features such as emulating the TCP stack, responding to huge number of requests, and providing flexible configuration on each service and virtual host.

Although honeyd is a very powerful honeypot, I would like to give some personal opinions:

Support ICMP emulation

Since the honeyd supports TCP and UDP responses emulation, it is also a good idea if honeyd can support ICMP responses emulation. For example, for ICMP subnet mask queries, the honeyd can response with a user-defined value in order to hide the actual network architecture.

Detail of logging information

The information shown from the honeyd does not provide details on the actual activities of the intruders. Although the logging can be handled by the emulation script, it is recommended that the complete commands/requests can be logged by honeyd using syslog (in remote server) for further analysis. This should be useful to understand the attack patterns performed by the intruder without the risk of being destroyed by the intruder.

Conclusion

There are several improvement areas on the existing honeypot such as remote logging for log protection, intrusion alerts for initiating responses and most important the TCP stack emulation. Without the TCP stack emulation, the honeypot may show inconsistent information when intruder tried to map and gather the target network information. This will raise suspicions to the intruder and subsequently no intruder will access the honeypot anymore. Without the intrusion information gathered, the honeypot is useless.

References

1. R Even, Loras. “What is a honeypot?” SANS FAQ. July 12, 2000 http://www.sans.org/newlook/resources/IDFAQ/honeypot3.htm (10 July, 2002)

2. Dostoyevsky, Fyodor. “Remote OS detection via TCP/IP Stack Fingerprinting.” June 11, 2002 http://www.insecure.org/nmap/nmap-fingerprinting-article.html (13 July, 2002)

3. Schwartau, Winn. “Lying to hackers is okay by me: Part 5 of 9.” 23 June, 1999 http://www.nwfusion.com/newsletters/sec/0621sec2.html (19 July, 2002)

4. Provos, Niels. “Honeyd” 30 July, 2002 http://www.citi.umich.edu/u/provos/honeyd/ (19 July, 2002)

5. Amoroso, Edward. Intrusion Detection – An introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response (First Edition). Intrusion.Net Books, 1999

6. Spitzner, Lance. “Honeypots – Definitions and Value of Honeypots” 17 May, 2002 http://www.enteract.com/~lspitz/honeypot.html (15 July, 2002)

7. Dostoyevsky, Fyodor. “Nmap” http://www.insecure.org/nmap (15 July, 2002)

END OF ASSIGNMENT 1

ASSIGNMENT 2 – NETWORK DETECTS

Network Detect #1 – DNS named version information leakage

Event Traces

The following is alerts generated by Snort v.1.8.7 (Build 128):

```
[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/17-03:29:02.754488 203.122.47.137:30930 -> XXX.YYY.17.139:53
UDP TTL:42 TOS:0x0 ID:55905 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/17-03:41:03.414488 203.122.47.137:20324 -> XXX.YYY.105.50:53
UDP TTL:42 TOS:0x0 ID:3114 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Pri ority: 2]
07/17-03:49:15.314488 203.122.47.137:28082 -> XXX.YYY.65.221:53
UDP TTL:40 TOS:0x0 ID:12922 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Infor mation Leak] [Priority: 2]
07/17-03:51:32.314488 203.122.47.137:30155 -> XXX.YYY.147.170:53
UDP TTL:40 TOS:0x0 ID:15596 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/17-04:12:09.304488 203.122.47.137:27647 -> XXX.YYY.63.240:53
UDP TTL:40 TOS:0x0 ID:40427 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

[**] [1:1616:1] DNS named version attempt [* *]
[Classification: Attempted Information Leak] [Priority: 2]
07/17-04:45:38.804488 203.122.47.137:15307 -> XXX.YYY.98.102:53
UDP TTL:42 TOS:0x0 ID:12990 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

<SKIPPED>

[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/17-05:10:02.034488 203.122.47.137:16374 -> XXX.YYY.28.14:53
UDP TTL:42 TOS:0x0 ID:41084 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

<SKIPPED>
```

```
[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/17-06:39:49.464488 203.122.47.137:12730 -> XXX.YYY.73.144:53
UDP TTL:42 TOS:0x0 ID:16483 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/17-06:41:39.364488 203.122.47.137:14485 -> XXX.YYY.42.246:53
UDP TTL:42 TOS:0x0 ID:18373 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

<SKIPPED>

[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/17-08:21:30.204488 203.122.47.137:20625 -> XXX.YYY.200.151:53
UDP TTL:40 TOS:0x0 ID:12942 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

<SKIPPED>

[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/17-08:48:30.134488 203.122.47.137:24069 -> XXX.YYY.163.85:53
UDP TTL:42 TOS:0x0 ID:52591 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

<SKIPPED>

[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/17-09:49:26.114488 203.122.47.137:15241 -> XXX.YYY.173.167:53
UDP TTL:42 TOS:0x0 ID:23775 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/17-10:27:01.154488 203.122.47.137:28719 -> XXX.YYY.64.70:53
UDP TTL:42 TOS:0x0 ID:13001 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

<SKIPPED>

[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Prior ity: 2]
07/17-11:01:07.154488 203.122.47.137:16978 -> XXX.YYY.37.33:53
UDP TTL:42 TOS:0x0 ID:52489 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

<SKIPPED>

[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
```

```
07/17-11:24:44.974488 203.122.47.137:17357 -> XXX.YYY.45.158:53
UDP TTL:42 TOS:0x0 ID:17879 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/17-11:58:43.434488 203.122.47.137:27361 -> XXX.YYY.169.184:53
UDP TTL:42 TOS:0x0 ID:54862 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

[**] [1:1616:1] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/17-12:36:05.394488 203.122.47.137:18639 -> XXX.YYY.236.155:53
UDP TTL:42 TOS:0x0 ID:28718 IpLen:20 DgmLen:58 Len: 38
[Xref => http://www.whitehats.com/info/IDS278]

<SKIPPED>
```

The following is generated by TCPdump using same log file:

```
03:29:02.754488 203.122.47.137.30930 > XXX.YYY.17.139.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
<SKIPPED>
03:41:03.414488 203.122.47.137.20324 > XXX.YYY.105.50.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
03:49:15.314488 203.122.47.137.28082 > XXX.YYY.65.221.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
03:51:32.314488 203.122.47.137.30155 > XXX.YYY.147.170.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
04:12:09.304488 203.122.47.137.27647 > XXX.YYY.63.240.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
04:45:38.804488 203.122.47.137.15307 > XXX.YYY.98.102.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
<SKIPPED>
05:10:02.034488 203.122.47.137.16374 > XXX.YYY.28.14 .53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
<SKIPPED>
06:39:49.464488 203.122.47.137.12730 > XXX.YYY.73.144.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
06:41:39.364488 203.122.47.137.14485 > XXX.YYY.42.246.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
<SKIPPED>
08:21:30.204488 203.122.47.137.20625 > XXX.YYY.200.151.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
<SKIPPED>
08:48:30.134488 203.122.47.137.24069 > XXX.YYY.163.85.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
<SKIPPED>
09:49:26.114488 203.122.47.137.15241 > XXX.YYY.173.167.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
10:27:01.154488 203.122.47.137.28719 > XXX.YYY.64.70.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
<SKIPPED>
11:01:07.154488 203.122.47.137.16978 > XXX.YYY.37.33.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
```

```
<SKIPPED>
11:24:44.974488 203.122.47.137.17357 > XXX.YYY.45.158.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
11:58:43.434488 203.122.47.137.27361 > XXX.YYY.169.184 .53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
12:36:05.394488 203.122.47.137.18639 > XXX.YYY.236.155.53:  4660
[b2&3=0x80] TXT CHAOS)? version.bind. (30)
<SKIPPED>
```

WHOIS Query

```
Query the APNIC Whois Database [whois.apnic.net node-2]

inetnum:      203.122.0.0 - 203.122.63.255
netname:      SPECTRANET
descr:        SPECTRA NET LIMITED
descr:        FIRST FIBRE BROADBAND NETWORK IN NEW DELHI,
INDIA.
country:      IN
admin-c:      UP1-AP
tech-c:       UP1-AP
mnt-by:       APNIC-HM
mnt-lower:    MAINT-IN-SPECTRA-NET-LTD
changed:      hostmaster@apnic.net 20000504
status:       ALLOCATED PORTABLE
source:       APNIC

person:       Uday Punj
address:      17-18, Nehru Place
address:      New Delhi - 110019
address:      India
country:      IN
phone:        +91-11-6200123
fax-no:       +91-11-6200111
e-mail:       sachin.mehra@in.spectranet.com
nic-hdl:      UP1-AP
mnt-by:       MAINT-NEW
changed:      gaurav.gulati@in.spectranet.com 20001205
source:       APNIC
```


Source of Trace

These packet traces are extracted from http://www.incidents.org/logs/Raw/2002.6.17 (Note that although the log filename is 2002.6.17, the log entries are 2002.7.17)

Detect was Generated by

The detect is generated by Snort IDS v.1.8.7 (Build 128) Linux Version with the Snort v.1.8.6 Ruleset. The exact filter that detected this kind of event is located in the ”dns.rules”.

Here is the filter that trigger to alert the event:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS named
version attempt"; content:"|07|version"; offset:12;
content:"|04|bind"; nocase; offset: 12;
reference:arachnids,278; classtype:attempted-recon; sid:1616;
rev:1;)
```

The alert was generated based on the following matches:

- UDP packet;
- The traffic flow from external network to Internal home network;
- Destination port is 53 (UDP);
- The 13th bytes (offset 12) of the pocket contents is a Hex value of "07" and then followed by a text "version" and then followed by a Hex value of "04" and text "bind".

The logs generated by SNORT 1.8.7 (Build 128):

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/17-03:29:02.754488 203.122.47.137:30930 -> XXX.YYY.17.139:53
UDP TTL:42 TOS:0x0 ID:55905 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4..........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....

<SKIPPED>

07/17-03:41:03.414488 203.122.47.137:20324 -> XXX.YYY.105.50:53
UDP TTL:42 TOS:0x0 ID:3114 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4..........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....

<SKIPPED>

07/17-03:49:15.314488 203.122.47.137:28082 -> XXX.YYY.65.221:53
UDP TTL:40 TOS:0x0 ID:12922 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4..........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/17-03:51:32.314488 203.122.47.137:30155 -> XXX.YYY.147.170:53
UDP TTL:40 TOS:0x0 ID:15596 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4..........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

<All others are the similar, so SKIPPED>
```

The above probe can be easily launched by using the “dig” or “nslookup” program. By using a script with target hosts’ IP addresses defined, the probe can be launched automatically. Here is the simple demonstration on using the “dig” in a RedHat Linux:

```
[root@gw log17]# dig @ns1.<DOMAIN>.com version.bind chaos
txt

; <<>> DiG 9.2.0 <<>> @ns1.<DOMAIN>.com version.bind chaos
txt
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37635
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 0

;; QUESTION SECTION:
;version.bind.                  CH      TXT

;; ANSWER SECTION:
VERSION.BIND.           0       CH      TXT     "8.2.3-REL"

;; Query time: 34 msec
;; SERVER: 205.252.XXX.YYY#53(ns1.<DOMAIN>.com)
;; WHEN: Tue Aug 27 21:05:52 2002
;; MSG SIZE  rcvd: 64
```

The snort in my machine also captured the packets that are closely matched with those in 2002.6.17:

```
09/01-16:45:30.092628 203.218.BB.AA:33015 -> XXX.YYY.ZZZ.228:53
UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:58 DF
Len: 38
94 EC 01 00 00 01 00 00 00 00 00 00 07 76 65 72  .............ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

09/01-16:45:30.112937 XXX.YYY.ZZZ.228:53 -> 203.218.BB.AA:33015
UDP TTL:252 TOS:0x0 ID:33599 IpLen:20 DgmLen:92 DF
Len: 72
94 EC 85 80 00 01 00 01 00 00 00 00 07 76 65 72  .............ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03 07 56  sion.bind......V
45 52 53 49 4F 4E 04 42 49 4E 44 00 00 10 00 03  ERSION.BIND.....
00 00 00 00 00 0A 09 38 2E 32 2E 33 2D 52 45 4C  .......8.2.3-REL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

This alert indicates a probe to determine the version of BIND running on the remote host.

Probability the Source Address was Spoofed

Low, although the UDP packets can be easily spoofed. With the SendIP command line utility developed by Mike Ricketts (or other generic packet generatoin tools like hping), we can specify the content of every header of a RIP, TCP, UDP, ICMP or raw IPv4 and IPv6 packet. For details, please refer to the URL http://www.earth.li/projectpurple/ppl/mike.html.

On the other hand, even though the full TCPDump log and syslog are not available for full analysis, I believe that the probability is low because the attacker wants to know the response from the remote hosts, if BIND DNS server is running. However, since there are no responses from the servers (maybe due to the log file are filtered or really no responses from the servers), we cannot definitely conclude that the source address was not spoofed.

Even though the source address was spoofed, the packet returned will not sent back to the attacker's machine, unless the attacker's machine is one of the node in the packet return paths. If this is the case, the attacker can hide his actual IP address and the reconnaissance cannot be easily discovered.

Description of the Attack

There are quite numbers of such alerts within the same day and some of the these queries are sent from other hosts which only be logged once. I tried to focus on the extracted log entries rather than the single alert because it is comparatively more interesting for further analysis.

The attacker is scanning to determine the version of BIND running on the network XXX.YYY.AAA.BBB. This appears to be a reconnaissance exercise. This is usually a pre-attack probe in order to gather more information for accurate exploitation on the BIND.

Attack Mechanism

The attack mechanism is that the attacker tries to perform a DNS query to determine the BIND version that is running on the remote hosts. According to the snort logs provided, the remote host 203.122.47.137 seems to be performing a *BIND version queries* to the hosts XXX.YYY.17.139, XXX.YYY.105.50, XXX.YYY.65.221, XXX.YYY.63.240 and other hosts within the network XXX.YYY.AAA.BBB. However, the probe is not in a sequential manner.

After gathered the BIND version number, the attacker may apply the corresponding "remote root compromise" techniques to launch the exploitation, if applicable, on the vulnerable DNS server.

There are quite a lot of tools publicly available on the BIND exploitation, however, I am not going to mention how to get the tools in this paper. Here is the extracted source code of the tools that used to exploit the BIND DNS server:

```
----------------
struct target_type
{
  char          desc[40];
  int           systype;
  unsigned long addr;
  unsigned long opt_addr;
  int           fd;
};

struct target_type target[] =
{
  {"x86 Linux 2.0.x named 4.9.5 -REL (se)",0,0xbffff21c,0xbffff23c,4},
  {"x86 Linux 2.0.x named 4.9.5 -REL (le)",0,0xbfffeedc,0xbfffeefc,4},
  {"x86 Linux 2.0.x named 4.9.5 -P1 (se)",0,0xbffff294,0xbffff2cc,4},
  {"x86 Linux 2.0.x named  4.9.5-P1 (le)",0,0xbfffef8c,0xbfffefb4,4},
  {"x86 Linux 2.0.x named 4.9.6 -REL (se)",0,0xbffff3e3,0xbffff403,4},
  {"x86 Linux 2.0.x named 4.9.6 -REL (le)",0,0xbffff188,0xbffff194,4},
  {"x86 Linux 2.0.x named 8.1 -REL (se)",0,0xbffff6a4,0xbffff6f8,5},
  {"x86 Linux 2.0.x named 8.1 -REL (le)",0,0xbffff364,0xbffff3b8,5},
  {"x86 Linux 2.0.x named 8.1.1 (se)",0,0xbffff6b8,0xbffff708,5},
  {"x86 Linux 2.0.x named 8.1.1 (le)",0,0xbffff378,0xbffff3c8,5},
  {"x86 FreeBSD 3.x named 4.9.5 -REL (se)",1,0xefbfd260,0xefbf d2c8,4},
  {"x86 FreeBSD 3.x named 4.9.5 -REL (le)",1,0xefbfd140,0xefbfd1a8,4},
  {"x86 FreeBSD 3.x named 4.9.5 -P1 (se)",1,0xefbfd260,0xefbfd2c8,4},
  {"x86 FreeBSD 3.x named 4.9.5 -P1 (le)",1,0xefbfd140,0xefbfd1a8,4},
  {"x86 FreeBSD 3.x named 4.9.6 -REL (se)",1,0xefbfd480,0xefbfd4e8,4},
  {"x86 FreeBSD 3.x named 4.9.6 -REL (le)",1,0xefbfd218,0xefbfd274,4},
  {{0},0,0,0,0}
};
------------------
```

I would call this as reconnaissance or probe, because the attacker is querying the BIND version number without actual exploitation packets. If the BIND version number returned from the remote hosts matched one of the above version (see the extracted program source code), the attacker can immediately execute this program to exploit the vulnerable DNS server.

Correlation

This detect is not new and corresponding CVE number is CVE-1999-0009.

http://ciac.llnl.gov/ciac/bulletins/k-050.shtml
http://www.securiteam.com/unixfocus/3Z5Q2Q0Q0C.html

Evidence of Active Targeting

As mentioned in the **Attack Mechanism** section, the attacker focused on the remote hosts in the network XXX.YYY.AAA.BBB. It seems not to be a general scan of an

entire network because the snort did not generate similar alerts related to all the hosts on the network. However, within the 3am – 12pm that day, the same source address probed 17 different hosts in the target network.

From one of the SNORT log entries shown below, the attacker directed this query specifically at a DNS server running BIND, not just any random machine.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/17-03:41:03.414488 203.122.47.137:20324 -> XXX.YYY.105.50:53
UDP TTL:42 TOS:0x0 ID:3114 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4..........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

It is quite obvious that this is active targeting.

I believe the attacker should have another prior reconnaissance before this one.
The following is the example of checking the DNS server of an domain using spectranet.COM (the ISP that the attacking source machine connected to):

```
----------
# dig spectranet.com
; <<>> DiG 9.2.0 <<>> spectranet.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59876
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4,
ADDITIONAL: 4

;; QUESTION SECTION:
;spectranet.com.                   IN      A

;; ANSWER SECTION:
spectranet.com.         22860   IN      A       203.122.63.152
spectranet.com.         22860   IN      A       203.122.63.154

;; AUTHORITY SECTION:
spectranet.com.         86351   IN      NS
ns.spectranet.com.
spectranet.com.         86351   IN      NS
NS5.spectranet.com.
spectranet.com.         86351   IN      NS
NS2.spectranet.com.
spectranet.com.         86351   IN      NS
ns4.spectranet.com.

;; ADDITIONAL SECTION:
ns.spectranet.com.      22366   IN      A       203.122.63.76
NS5.spectranet.com.     133344  IN      A       203.122.63.152
```

```
NS2.spectranet.com.     133344   IN      A       203.122.63.154
ns4.spectranet.com.     86351    IN      A       203.122.63.77
-----------
```

Then, if I were the attacker, I will target to query the version of the DNS on 202.122.63.76, .152, .154 and .77.

Severity

The following formula calculates the severity of the attack. The metrics are assigned on a five-point scale, with 5 being highest and 1 the lowest.

Target Criticality = 1
Since DNS Server is a critical component, this could have a great impact to communications between Internet and the organization. However, there are no return packets sent out from the target hosts. I assume these target hosts are not running a DNS daemon or they have already implemented other countermeasures such as ISS Server Sensors, which can filter such incoming packets.

Attack Lethality = 2
The attack is a reconnaissance scan.

System Countermeasures = 3
There is no return packets sent out from the target hosts. The packets maybe filtered by countermeasures, or the host is not running DNS daemon.

Network Countermeasures = 3
Since the target hosts maybe located in different subnets, firewall maybe in-place as we cannot see the return packets sent from the target hosts. The packet may not reach the target.

Severity = (Target Criticality + Attack Lethality) – (System Countermeasures + Network Countermeasures)
= (1 + 2) – (3 + 3)
= -3

Defensive Recommendation

There are several controls could be implemented in order to mitigate the risks of further actions taken by the attack:

1) As we cannot conclude the probe is finished, the attacker may probe all the hosts in the XXX.YYY.AAA.BBB network. It is recommended to immediately disable the DNS daemon in other hosts, if it is not absolutely necessary;
2) Install all the security patches that are available;
3) Disable zone transfer, or if really needed, only allow authorized host(s) to communicate with the DNS server using TCP/53.

4) Hide the version number of the BIND in reply by editing `/usr/src/contrib/bind/Version` and recompiling them.

Multiple Choice Question

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ =+=+=+

07/17-03:41:03.414488 203.122.47.137:20324 -> XXX.YYY.105.50:53
UDP TTL:42 TOS:0x0 ID:3114 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72  .4...........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03        sion.bind.....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Which of the following statement is true, if XXX.YYY.105.50 is running a BIND DNS Server?

A. The Source IP is spoofed in order to hide the actual host that initiating this attack.
B. The probe can be completely eliminated by implementing an ACL which block UDP/53 of the public DNS server.
C. The probe cannot be prevented because the DNS port (UDP/53) is required to by accessed by public.
D. The probe is successful and DNS zone as been transferred successfully.

Answer: C. The probe cannot be prevented but we can hide the version of the BIND by editing `/usr/src/contrib/bin/Version` and recompile for a new binary `named`.

Network Detect # 2 – SOCKS Proxy Scanning?

Event Traces

The following is alerts generated by Snort v.1.8.7 (Build 128):

```
[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/01-05:37:53.444488 195.119.1.180:4045 -> XXX.YYY.121.142:1080
TCP TTL:43 TOS:0x0 ID:61038 IpLen:20 DgmLen:60  DF
******S* Seq: 0xFD8D57B6  Ack: 0x0  Win: 0x7D78  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 2830520519 0 NOP WS: 0
[Xref => http://help.undernet.org/proxyscan/]

<SKIPPED>

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/01-22:00:39.174488 64.228.107.58:37917 -> XXX.YYY.173.229:1080
TCP TTL:50 TOS:0x0 ID:38916 IpLen:20 DgmLen:60 DF
******S* Seq: 0x535B4A2D  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 89313833  0 NOP WS: 0
[Xref => http://help.undernet.org/proxyscan/]

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/01-22:00:42.154488 64.228.107.58:37917 -> XXX.YYY.173.229:1080
TCP TTL:50 TOS:0x0 ID:38917 IpLen:20 DgmLen:60 DF
******S* Seq: 0x535B4A2D  Ack: 0x0  Win: 0x16D0  TcpLen: 40

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/01-22:00:52.504488 64.228.107.58:40241 -> XXX.YYY.173.229:1080
TCP TTL:50 TOS:0x0 ID:9868 IpLen:20 DgmLen:60 DF
******S* Seq: 0x53D5F09D  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 89314833 0 NOP WS: 0
[Xref => http://help.undernet.org/proxyscan/]

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/01-22:00:52.824488 64.228.107.58:40241 -> XXX.YYY.173.229:1080
TCP TTL:50 TOS:0x0 ID:9869 IpLen:20 DgmLen:60 DF
******S* Seq: 0x53D5F09D  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 89315133 0 NOP WS: 0
[Xref => http://help.undernet.org/proxyscan/]

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/01-22:00:58.164488 64.228.107.58:40241 -> XXX.YYY.173.229:1080
TCP TTL:50 TOS:0x0 ID:9870 IpLen:20 DgmLen:60 DF
******S* Seq: 0x53D5F09D  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 89315733 0 NOP WS: 0
[Xref => http://help.undernet.org/proxyscan/]

<SKIPPED as most of them are similar>
```

The following is generated by TCPdump using same log file:

```
05:37:53.444488 195.119.1.180.4045 > XXX.YYY.121.142.1080: S
4253898678:4253898678(0) win 32120 <mss 1460,sackOK,timestamp
2830520519 0,nop,wscale 0> (DF)
22:00:39.174488 64.228.107.58.37917 > XXX.YYY.173.229.1080: S
1398491693:1398491693(0) win 5840 <mss 1460,sackOK,timestamp
89313833 0,nop,wscale 0> (DF)
22:00:42.154488 64.228.107.58.37917 > XXX.YYY.173.229.1080: S
1398491693:1398491693(0) win 5840 <mss 1460,sackOK,timestamp
89314133 0,nop,wscale 0> (DF)
22:00:48.174488 64.228.107.58.37917 > XXX.YYY.173.229.1080: S
1398491693:1398491693(0) win 5840 <mss 1460,sackOK,timestamp
89314733 0,nop,wscale 0> (DF)
22:00:52.504488 64.228.107.58.40241 > XXX.YYY.173.229.1080: S
1406529693:1406529693(0) win 5840 <mss 1460,sackOK,timestamp
89314833 0,nop,wscale 0> (DF)
22:00:52.824488 64.228.107.58.40241 > XXX.YYY.173.229.1080: S
1406529693:1406529693(0) win 5840 <mss 1460,sackOK,timestamp
89315133 0,nop,wscale 0> (DF)
22:00:58.164488 64.228.107.58.40241 > XXX.YYY.173.229.1080: S
1406529693:1406529693(0) win 5840 <mss 1460,sackOK,timestamp
89315733 0,nop,wscale 0> (DF)

<SKIPPED as most of them are similar>
```

WHOIS Query

The WHOIS Query is obtained from www.arin.net (American Registry for Internet Numbers).

Search results for: NETBLK-SYMP20002-CA

```
Sympatico (NETBLK-SYMP20002-CA)
   76 Adelaide
   Toronto, Ontario M5H 1P6
   CA

   Netname: SYMP20002-CA
   Netblock: 64.228.96.0 - 64.228.127.255

   Coordinator:
      Daoust, Philippe  (PD135-ARIN)  noc@in.bell.ca
      +1-800-450-7771 +1-416-215-5423

   Record last updated on 18-Apr-2000.
   Database last updated on  22-Aug-2002 22:40:20 EDT.
```

Source of Trace

These packet traces are extracted from http://www.incidents.org/logs/Raw/2002.6.1 (Note that although the log filename is 2002.6.17, the log entries are 2002.7.17)

Detect was Generated by

The detect is generated by Snort IDS v.1.8.7 (Build 128) Linux Version with the Snort v.1.8.6 Ruleset. The exact filter that detected this kind of event is located in the "scan.rules".

Here is the filter that trigger to alert the event:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 1080 (msg:"SCAN SOCKS
Proxy attempt"; flags:S;
reference:url,help.undernet.org/proxyscan/;
classtype:attempted-recon; sid:615; rev:3;)
```

The alert was generated based on the following matches:
1) TCP packet;
2) The traffic flow from external network to Internal home network;
3) Destination port is 1080 (TCP);
4) The SYN flag is set.

The logs generated by SNORT 1.8.7 (Build 128):

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/01-22:00:39.174488 64.228.107.58:37917 -> XXX.YYY.173.229:1080
TCP TTL:50 TOS:0x0 ID:38916 IpLen:20 DgmLen:60 DF
******S* Seq: 0x535B4A2D  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 89313833 0 NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/01-22:00:42.154488 64.228.107.58:37917 -> XXX.YYY.173.229:1080
TCP TTL:50 TOS:0x0 ID:38917 IpLen:20 DgmLen:60 DF
******S* Seq: 0x535B4A2D  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 89314133 0 NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/01-22:00:48.174488 64.228.107.58:37917 -> XXX.YYY.173.229:1080
TCP TTL:50 TOS:0x0 ID:38918 IpLen:20 DgmLen:60 DF
******S* Seq: 0x535B4A2D  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 89314733 0 NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/01-22:00:52.504488 64.228.107.58:40241 -> XXX.YYY.173.229:1080
TCP TTL:50 TOS:0x0 ID:9868 IpLen:20 DgmLen:60 DF
******S* Seq: 0x53D5F09D  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 89314833 0 NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/01-22:00:52.824488 64.228.107.58:40241 -> XXX.YYY.173.229:1080
TCP TTL:50 TOS:0x0 ID:9869 IpLen:20 DgmLen:60  DF
```

```
******S* Seq: 0x53D5F09D  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 89315133 0 NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

<SKIPPED as most of them are similar>
```

This alert indicates a probe to determine the availability of a SOCK server on the remote host.

Probability the Source Address was Spoofed

Since the SYN flag is set, so there is a probability that the source address was spoofed if this is a SYN flood attack. However, according to the time of each packet sent, each SYN request is few seconds before next request, so probably it is not a SYN flood attack. So the source address is not necessary being spoofed.

Besides, there is no sign of packet crafting because the source IP address, sequence no. and acknowledgement no. seem normal.

Description of the Attack

SOCKS is a system that allows multiple machines to share a common Internet connection. SOCKS Proxy Scanning attempts to locate the machine that has SOCKS Proxy Server running by connecting to the SOCKS Server default port TCP/1080.

However, according to the log entries for whole day (2002/6/1), I would classify this as false positive because:

1) the "attacker's machine" tried to connect to one SOCKS Server only, without attempting to connect to other machines in the networks.
2) the packet resent few times for every few seconds (same Sequence No.), which should be a normal behavior of retry of sending TCP packet.
3) the source address seems to be an ISP IP address that assigned to their dial-up customer.

```
C:\>nslookup 64.228.107.58
Server:  dns02.netvigator.com
Address:  208.167.231.55

Name:    Toronto-ppp221495.sympatico.ca
Address:  64.228.107.58
```

Attack Mechanism

Since SOCKS Proxy Server is not picky about the source and destination, it originally designs for allowing the internal machines to share the common Internet

connection. With this reasons, it will also allow Internet machines to access the internal network.

With the characteristic of the Proxy, it allows attackers to hide their actual IP by using the vulnerable / misconfigured SOCKS Proxy Server.

Some of the IRC networks, such as undernet.org, will scan the user whenever they connect, in order to find out the potentially vulnerable proxy servers. If such a server is found, then its access to the IRC network is denied. However, sometimes the scanning is initiated by attackers or script kiddies.

The URL http://www.fr1.cyberabuse.org/?page=abuse-proxy describes the details of abuse of proxy. The attacker may try to scan for vulnerable proxy servers by performing a large scale port scanning on remote hosts port TCP/1080.

The attack mechanism is that the "attacker" tries to initiate a connection to a machine on the remote network, in order to determine whether a SOCKS server is running. Assume it is a real attack, it should be a reconnaissance exercise, There are quite a lot of scanning techniques such as SYN Scan, FIN Scan, ACK Scan and others, for scanning the active ports listening on a server. Again, if it was a real attack, then it would be classified as SYN Scan.

According to the snort logs provided, the remote host 64.228.107.58 perform the connection requests to XXX.YYY.173.229 on port 1080. However, this is false positive (refer to Description of the Attack).

Correlations

Since this is a false positive, there is no correlations can be found.

Evidence of Active Targeting

Since this is a false positive, I would treat it as misconfiguration instead of attack. If it was a real attack, it would be considered as active targeting because It seems not to be a general scan of an entire network.

Severity

The following formula calculates the severity of the attack. The metrics are assigned on a five-point scale, with 5 being highest and 1 the lowest.

Target Criticality = 3
SOCKS may be a critical component because it shares a common Internet connection for the whole (internal) network. This could have a significant impact to communications between Internet and the organization.

Attack Lethality = 1

Since, it is a false positive so the lethality is considered to be 1. If it was a real attack, the attack is a reconnaissance scan so the lethality may be considered as 2 then.

System Countermeasures = 3
Traffic maybe allowed to the target but there was no response from the target, according to the traffic log. Either there is an countermeasure in place to protect the target, or SOCKS daemon is not being run in the target. As there is no sufficient information, we cannot conclude the system countermeasures that have been implemented, and thus I would classify it as 3.

Network Countermeasures = 3
Patches installed or not cannot be determined. Implementation of stateful firewall can prevent the incoming SOCKS connection initiating from external networks. Host-based IDS may also catch the packet and give protective responses on the requests. Again, as there is no sufficient information, we cannot conclude the network countermeasures that have been implemented, and thus I would classify it as 3.

Severity = (Target Criticality + Attack Lethality) – (System Countermeasures + Network Countermeasures)
= (3 + 1) – (3 + 3)
= -2

Defensive Recommendation

There are several controls could be implemented in order to mitigate the risks of further actions taken by the attack:

1) Disable the SOCKS daemon if it is not absolutely necessary;
2) Install all the security patches that are available;
3) Implement the stateful firewall policy so that incoming TCP/1080 is denied;
4) To further avoid the abuse of the SOCKS, the SOCKS connections should only be allowed upon successful authentication.

Multiple Choice Question

Look at the following packet trace of “Scan SOCKS Proxy attempt” alert. Is this a real attack or not?

```
22:00:39.174488 64.228.107.58.37917 > XXX.YYY.173.229.1080: S
1398491693:1398491693(0) win 5840 <mss 1460,sackOK,timestamp
89313833 0,nop,wscale 0> (DF)
22:00:42.154488 64.228.107.58.37917 > XXX.YYY.173.229.1080: S
1398491693:1398491693(0) win 5840 <mss 1460,sackOK,timestamp
89314133 0,nop,wscale 0> (DF)
22:00:48.174488 64.228.107.58.37917 > XXX.YYY.173.229.1080: S
1398491693:1398491693(0) win 5840 <mss 1460,sackOK,timestamp
89314733 0,nop,wscale 0> (DF)
```

```
22:00:52.504488 64.228.107.58.40241 > XXX.YYY.173.229.1080: S
1406529693:1406529693(0) win 5840 <mss 1460,sackOK,timestamp
89314833 0,nop,wscale 0> (DF)
22:00:52.824488 64.228.107.58.40241 > XXX.YYY.173.229.1080:  S
1406529693:1406529693(0) win 5840 <mss 1460,sackOK,timestamp
89315133 0,nop,wscale 0> (DF)
22:00:58.164488 64.228.107.58.40241 > XXX.YYY.173.229.1080: S
1406529693:1406529693(0) win 5840 <mss 1460,sackOK,timestamp
89315733 0,nop,wscale 0> (DF)
```

A. True positive, because the attacker continuously scans the target host using ACK Scan technique.
B. True positive, because the attacker generally scans the entire target network for SOCKS connection.
C. False positive, because of the misconfiguration of SOCKS configuration in the host 64.228.107.58.
D. False positive, because of the successful of the SOCKS connections between 64.228.107.58 and XXX.YYY.173.229.

Answer C.

Network Detect #3 – Bad Traffic TCP Port 0 Traffic

Event Traces

The following is alerts generated by Snort v.1.8.7 (Build 128):

```
[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
07/12-21:50:41.374488 211.47.255.24:45149 -> XXX.YYY.236.82:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xD56EB608  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
07/12-21:50:44.364488 211.47.255.24:45149 -> XXX.YYY.236.82:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xD56EB608  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
07/12-21:50:50.364488 211.47.255.24:45149 -> XXX.YYY.236.82:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xD56EB608  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
07/12-21:51:02.364488 211.47.255.24:45149 -> XXX.YYY.236.82:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xD56EB608  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
07/12-21:51:13.374488 211.47.255.24:45327 -> XXX.YYY.236.82:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xD81C4A4D  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

<SKIPPED as most of them are similar>

[**] [1:524:3] BAD TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
07/12-21:52:38.364488 211.47.255.24:45643 -> XXX.YYY.236.82:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xDC3741F9  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

<Total 16 Alerts within 2 minutes>
```

The following is generated by TCPdump using same log file:

```
21:50:41.374488 211.47.255.24.45149 > XXX.YYY.236.82.0: S
3580802568:3580802568(0) win 5840 <mss
1460,nop,nop,sackOK,nop,wscale 0> (DF)
21:50:44.364488 211.47.255.24.45149 > XXX.YYY.236.82.0: S
3580802568:3580802568(0) win  5840 <mss
1460,nop,nop,sackOK,nop,wscale 0> (DF)
21:50:50.364488 211.47.255.24.45149 > XXX.YYY.236.82.0: S
3580802568:3580802568(0) win 5840 <mss
1460,nop,nop,sackOK,nop,wscale 0> (DF)
21:51:02.364488 211.47.255.24.45149 > XXX.YYY.236.82.0: S
3580802568:3580802568(0) win 5840 <mss
1460,nop,nop,sackOK,nop,wscale 0> (DF)
21:51:13.374488 211.47.255.24.45327 > XXX.YYY.236.82.0: S
3625732685:3625732685(0) win 5840 <mss
1460,nop,nop,sackOK,nop,wscale 0> (DF)
<SKIPPED>
21:52:38.364488 211.47.255.24.45643 > XXX.YYY.236.82.0: S
3694608889:3694608889(0) win 5840 <mss
1460,nop,nop,sackOK,nop,wscale 0> (DF)
```

WHOIS Query

```
Query the APNIC Whois Database

inetnum:      211.47.255.0 - 211.47.255.255
netname:      ORG84651-KR
descr:        SAEROUNNET
descr:        789-28 sihungdong kumchungu
descr:        SEOUL
descr:        153-034
country:      KR
admin-c:      CK1008-KR
tech-c:       IS493-KR
remarks:      This IP address space has been allocated to KRNIC.
remarks:      For more information, using KRNIC Whois Database
remarks:      whois -h whois.nic.or.kr
mnt-by:       MNT-KRNIC-AP
remarks:      This information has been partially mirrored by APNIC
              from
remarks:      KRNIC. To obtain more specific information, please use
              the
remarks:      KRNIC whois server at whois.krnic.net.
changed:      hostmaster@nic.or.kr 20020826
source:       KRNIC

person:       Chang Kim
descr:        SAEROUNNET
descr:        789-28 sihungdong kumchungu
descr:        SEOUL
descr:        153-034
country:      KR
phone:        +82-17-334-8450
fax-no:       +82-2-836-0274
e-mail:       seesky@saeroun.co.kr
nic-hdl:      CK1008-KR
```

```
mnt-by:        MNT-KRNIC-AP
remarks:        This information has been partially mirrored by APNIC
                from
remarks:       KRNIC. To obtain more specific information, please use
                the
remarks:       KRNIC whois server at whois.krnic.net.
changed:       hostmaster@nic.or.kr 20020826
source:        KRNIC

person:        In suk Jung
descr:         SAEROUNNET
descr:         789-28 sihungdong kumchungu
descr:         SEOUL
descr:         153-034
country:       KR
phone:         +82-16-202-7956
fax-no:        +82-2-836-0274
e-mail:        ip@saeroun.co.kr
nic-hdl:       IS493-KR
mnt-by:        MNT-KRNIC-AP
remarks:        This information has been partially mirrored by APNIC
                from
remarks:       KRNIC. To obtain more specific information, please use
                the
remarks:       KRNIC whois server at whois.krnic.net.
changed:       hostmaster@nic.or.kr 20020826
source:        KRNIC
```

Source of Trace

The detect is generated by Snort IDS v.1.8.7 (Build 128) Linux Version with the snort v.1.8.6 Ruleset. The exact filter that detected this kind of event is located in the "bad-traffic.rules".

Here is the filter that trigger to alert the event:

```
alert tcp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD TRAFFIC tcp
port 0 traffic"; sid:524; classtype:misc-activity; rev:3;)
```

The alert was generated based on the following matches:
1) TCP packet;
2) The traffic flow from external network to Internal home network; and
3) Destination port is 0 (TCP);

The logs generated by SNORT 1.8.7 (Build 128):

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/12-21:50:41.374488 211.47.255.24:45149 -> XXX.YYY.236.82:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xD56EB608  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

```
07/12-21:50:44.364488 211.47.255.24:45149 -> XXX.YYY.236.82:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xD56EB608  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/12-21:50:50.364488 211.47.255.24:45149 -> XXX.YYY.236.82:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xD56EB608  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/12-21:51:02.364488 211.47.255.24:45149 -> XXX.YYY.236.82:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xD56EB608  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/12-21:51:13.374488 211.47.255.24:45327 -> XXX.YYY.236.82:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xD81C4A4D  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

<SKIPPED>

07/12-21:52:38.364488 211.47.255.24:45643 -> XXX.YYY.236.82:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0xDC3741F9  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Since Port 0 is officially a reserved port in TCP/IP networking, meaning that it should not be used for any TCP or UDP network communications. So this alert indicates an unusual traffic is detected.

Probability the Source Address was Spoofed

Since the SYN flag is set, so there is a probability that the source address was spoofed if this is a SYN flood attack. However, according to the time of each packet sent, each SYN request is few seconds before next request, so probably it is not a SYN flood attack. So the source address is not necessary being spoofed.

Assume the source address was spoofed, the packet returned will not sent back to the attacker's machine, unless the attacker's machine is one of the node in the packet return paths. If this is the case, the attacker can hide his actual IP address and the reconnaissance cannot be easily discovered.

On the other hand, the source IP address, sequence no. and acknowledgement no. seem normal, but other fields such as destination port and ID may be crafted.

By looking at the value of the ID field (0x0) of the packet, I have had wondered that ID field has been crafted. However, when looking at my demonstration of BIND version query using "dig" in Network Detect #1 above again, the ID field is also 0x0 and that was not crafted. So probably only "destination port" has been crafted.

Although there is no full TCPdump log for further analysis, with the above information, I believe the probability the source address was spoofed is low (around 20% probability that they were spoofed).

Description of the Attack

The attacker tried to send totally 16 packets from his/her machine (211.47.255.24) to the remote host (XXX.YYY.236.82) with destination port TCP/0, which is a reserved port. As some packets are having same Sequence No., Source Port No., and Destination Address, it is obvious that some of them are TCP resent packets due to expected acknowledgement cannot be received.

As there is no sufficient information and payload data can be found in the log, it is difficult to conclude what the function of the TCP/0 in the remote host was. As mentioned before, TCP/0 is a reserved port so I would assume the remote host is not running an daemon listening on TCP/0.

With this assumption, I would also assume that no application from external hosts will connect to TCP/0 of the remote host (XXX.YYY.236.82). The attacker may make use of scanning tool like "hping" to perform a reconnaissance exercise. By default, "hping" uses default port 0 unless the port is explicitly specified using the "–p <port>" parameter. (Hping Manpage: http://www.hping.org/manpage.html)

Attack Mechanism

The attack mechanism is that the attacker tried to perform an remote OS fingerprinting using packet generation tool like "hping". "Hping" uses default port 0 when performing a port scanning/OS fingerprinting process. As connecting to the active ports on the remote host may not give sufficient information by the responses, OS fingerprinting tool usually sends out-of-spec or specially crafted packets and then determine the OS due to the different implementation of exception handling. See my white paper (assignment 1) for more details.

Here is the "hping" command that I expected to do similar reconnaissance exercise:

```
# hping –S –p 0 –L 0 –s 45149 –w 5840 –y XXX.YYY.236.82
```

Correlation

There are not much similar detect reported. Here is just one I can find:

http://www.geocrawler.com/archives/3/6752/2002/3/0/8233030/

Evidence of Active Targeting

The attacker's host specifically "SYN Scan" the target host (XXX.YYY.236.82), so it is obvious that it is active targeting because it seems not to be a general scan of an entire network.

Severity

The following formula calculates the severity of the attack. The metrics are assigned on a five-point scale, with 5 being highest and 1 the lowest.

Target Criticality = 3
As there is no sufficient information on the target host, it may be a critical component or an non-existence host. Due to the uncertainty, the criticality would be classified as 3 (medium).

Attack Lethality = 2
It is a reconnaissance scan so the lethality may be considered as 2.

System Countermeasures = 3
Traffic maybe allowed to the target but there was no response from the target, according to the traffic log. Either there is an countermeasure in place to protect the target, or the target did not response to the TCP/0 connections. As there is no sufficient information, we cannot conclude the system countermeasures that have been implemented, and thus I would classify it as 3.

Network Countermeasures = 3
Patches installed or not cannot be determined. Implementation of stateful firewall can prevent the incoming TCP/0 connection initiating from external networks. Host-based IDS may also catch the packet and give protective responses on the requests. Again, as there is no sufficient information, we cannot conclude the network countermeasures that have been implemented, and thus I would classify it as 3.

Severity = (Target Criticality + Attack Lethality) – (System Countermeasures + Network Countermeasures)
= (3 + 2) – (3 + 3)
= -1

Defensive Recommendation

There are several controls could be implemented in order to mitigate the risks of further actions taken by the attack:

1) Implement firewall policy so that incoming TCP/0 is denied and logged;

2) Implement host-based IDS so that unusual traffic is logged and dropped, such that even such reconnaissance exercise is performed within your internal network with no firewall in place; and
3) Perform OS hardening (and TCP wrapper, for UNIX host) so that only authorized host(s) can connect to TCP/0 (if absolutely needed).

Multiple Choice Question

Which of the following snort log is correct, if the following command is used to perform remote OS fingerprinting?

```
# hping –S –p 0 –L 0 –s 45149 –w 5840 –y XXX.YYY.236.82
```

A.

```
09/02-19:57:55.378606 my.net:45149 -> XXX.YYY.236.82:0
TCP TTL:64 TOS:0x0 ID:60854 IpLen:20 DgmLen:40 DF
******S* Seq: 0x4D4C1119  Ack: 0x0  Win: 0x16D0  TcpLen: 20
```

B.

```
09/02-19:59:05.508637 my.net:45149 -> XXX.YYY.236.82:20
TCP TTL:64 TOS:0x8 ID:20364 IpLen:20 DgmLen:40 DF
******S* Seq: 0x7F856E5A  Ack: 0x0  Win: 0x16D0  TcpLen: 20
```

C.

```
09/02-20:01:46.418619 my.net 16:1104 -> XXX.YYY.236.82:0
TCP TTL:64 TOS:0x0 ID:49429 IpLen:20 DgmLen:40
******S* Seq: 0x6608038E  Ack: 0x2B140683  Win: 0x200  TcpLen: 20
```

D.

```
09/02-20:03:41.418781 my.net:1104 -> XXX.YYY.236.82:0
TCP TTL:64 TOS:0x0 ID:50436 IpLen:20 DgmLen:40 DF
******S* Seq: 0x6601451E  Ack: 0x0  Win: 0x200  TcpLen: 20
```

Answer: A

B is wrong because the destination port is not 0.
C and D is wrong because the source port is not 45149, Windows Size is not 5840 and similar mismatch values.

ASSIGNMENT 3 - ANALYSE THIS

Executive Summary

A network security audit was performed by analyzing different types of logs provided, which contains 5 consecutive days data. There are 80 unique attacks over the 253,685 alerts generated by the Snort IDS.

During the analysis, it was discovered that there are huge amount of traffic using 1214/tcp and 1214/udp within the network. The traffic not only between the hosts in the internal network, but also involves external hosts. These 1214/tcp and 1214/udp services should be used by a MP3 file-sharing program called KaZaA or Morpheus.

However, other backdoor Trojan program may also use these services ports for remote controlling the infected hosts. This kind of traffic should be monitored closely, especially the traffic coming from external hosts. The suspicious traffic has been highlighted in the "Top 5 High Severity Alerts #1 - Watchlist 000220 IL-ISDNNET-990517" section in this report for immediately further investigation.

Other than that, there are quite a number of false alarms triggered and reconnaissance traffic detected.

There was no host found to be compromised definitely. However, further investigation on the communication between external hosts and internal hosts using 1214/tcp and 1214/udp should be taken.

In summary, the network is under controlled, but it is recommended that the network and system administrators should consider to harden or fine-tune the parameters of the systems such as changing the default SNMP community string, installing latest security patches, implement basic ACLs in the router to prevent IP spoofing and packet fragmentation.

Details of the Analysis

List of the Files Analysed

During this security audit, several Snort alerts files are obtained from http://www.incidents.org/logs/ for analysis. Here are the lists of the files:

Filename	Size (Bytes)	Date
alert.020601.gz	1,315,059	Mon Jun 17 18:24:59 2002
alert.020602.gz	1,495,773	Mon Jun 17 18:24:59 2002
alert.020603.gz	2,209,151	Mon Jun 17 18:25:00 2002
alert.020604.gz	2,129,106	Mon Jun 17 18:25:00 2002
alert.020605.gz	2,159,472	Mon Jun 17 18:25:01 2002
scans.020601.gz	5,126,395	Mon Jun 17 18:26:03 2002

scans.020602.gz	5,949,588	Mon Jun 17 18:26:04 2002
scans.020603.gz	6,060,686	Mon Jun 17 18:26:06 2002
scans.020604.gz	7,196,698	Mon Jun 17 18:26:08 2002
scans.020605.gz	5,628,730	Mon Jun 17 18:26:09 2002
oos_logs_jun_2002.tar.gz	101,040	Mon Jun 24 16:07:09 2002

As there are several OOS log files contained in the oos_logs_jun_2002.tar.gz, only 5 OOS logs that close to the Scan files and Alert files have been analysed. Here are the lists of the OOS files analysed:

Filename	Size (Bytes)
OOS_Jun.3.2002	11,655
OOS_Jun.5.2002	3,031
OOS_Jun.12.2002	266
OOS_Jun.14.2002	2790
OOS_Jun.15.2002	846

As the alert files contained the port scanning information, they are ignored during the analysis. There are 253,685 alerts during these 5 days.

The alert files were unzipped to a temporary directory and then concatenated to a single alert file. After that, the alert file was filtered out the port scan alerts using the following command:

```
# cat alert.log | grep –v "spp_portscan" > alert.log.filtered
```

In order to easily parse and import it into the database, the filtered alert file was edited to replace "[**]" and "->" by a delimiter "|" using the following command:

```
# cat alert.log.filtered | sed 's/\[\*\*\]/\|/g' > alert.log.changed
# cat alert.log.changed | sed 's/->/\|/g' > alert.log.final
```

The alert.log.final was then import to the MS SQL database (in Windows 2000 platform) for further analysis using SQL statements.

Similar to the alert files, the port scan files were unzipped to a temporary directory, concatenated, parsed and edited to an "importable" format for importing into the database for further analysis. There are 4,174,129 port scan alerts during these 5 days. (Details of the analysis processes, please refer to last section of this assignment – Analysis Processes)

Analysis of Alerts Data

By using several simple SQL statements to query the log entries obtained, the table containing the alerts triggered during these 5 days, sorted by occurrence, was produced as below.

Alerts Statistics with traffic direction breakdown

Alerts Triggered	# Alerts	%	Ext->Ext	Ext->Int	Int->Ext	Int->Int
SMB Name Wildcard	50,770	20.01%		28		50742
spp_http_decode: IIS Unicode attack detected	43,639	17.20%		2,995	40,634	10
SNMP public access	37,472	14.77%				37472
MISC Large UDP Packet	31,877	12.57%		31,877		
ICMP Echo Request L3retriever Ping	23,784	9.38%		27		23757
connect to 515 from inside	21,162	8.34%				21162
Watchlist 000220 IL-ISDNNET-990517	11,112	4.38%		11,112		
spp_http_decode: CGI Null Byte attack detected	7,585	2.99%			7,584	1
INFO MSN IM Chat data	6,951	2.74%		3,491	3,460	
ICMP Echo Request Nmap or HPING2	3,553	1.40%			7	3546
High port 65535 udp - possible Red Worm - traffic	3,052	1.20%		580		2472
WEB-MISC Attempt to execute cmd	2,299	0.91%		2,299		
FTP DoS ftpd globbing	2,203	0.87%		2,203		
INFO Inbound GNUTella Connect request	1,381	0.54%		1,381		
Incomplete Packet Fragments Discarded	1,078	0.42%		1,075	3	
ICMP Router Selection	848	0.33%			848	
ICMP Fragment Reassembly Time Exceeded	766	0.30%			746	20
INFO Outbound GNUTella Connect request	658	0.26%			658	
Null scan!	542	0.21%		541		1
High port 65535 tcp - possible Red Worm - traffic	516	0.20%		189	325	2
ICMP Echo Request Windows	304	0.12%		21	185	98
WEB-IIS view source via translate header	288	0.11%		288		
SCAN Proxy attempt	226	0.09%		226		
Watchlist 000222 NET-NCFC	176	0.07%		176		
ICMP Destination Unreachable (Communication Administratively Prohibited)	172	0.07%				172
TCP SRC and DST outside network	140	0.06%	140			
UDP SRC and DST outside network	107	0.04%	107			
INFO Possible IRC Access	95	0.04%			95	
WEB-MISC 403 Forbidden	87	0.03%			87	
INFO FTP anonymous FTP	79	0.03%		79		
ICMP Echo Request CyberKit 2.2 Windows	76	0.03%			76	
ICMP traceroute	61	0.02%			3	58
WEB-FRONTPAGE _vti_rpc access	60	0.02%		60		
WEB-IIS _vti_inf access	60	0.02%		60		
ICMP Echo Request BSDtype	47	0.02%				47
WEB-IIS Unauthorized IP Access Attempt	39	0.02%			39	
SCAN Synscan Portscan ID 19104	24	0.01%		24		
EXPLOIT x86 setuid 0	24	0.01%		24		
WEB-MISC whisker head	23	0.01%		23		
WEB-MISC http directory traversal	22	0.01%		22		
SYN-FIN scan!	21	0.01%		21		
INFO - Possible Squid Scan	20	0.01%		20		

TFTP - Internal UDP connection to extern al tftp server	19	0.01%		2	17	
Port 55850 tcp - Possible myserver activity - ref. 010313-1	19	0.01%		9	8	2
RFB - Possible WinVNC - 010708-1	18	0.01%		5		13
EXPLOIT NTPDX buffer overflow	17	0.01%		17		
x86 NOOP - unicode BUFF ER OVERFLOW ATTACK	16	0.01%		16		
WEB-IIS 5 .printer isapi	14	0.01%		14		
NMAP TCP ping!	14	0.01%		12		2
IDS552/web-iis_IIS ISAPI Overflow ida nosize	14	0.01%		14		
Queso fingerprint	12	0.00%		12		
EXPLOIT x86 NOOP	12	0.00%		12		
EXPLOIT x86 setgid 0	12	0.00%		12		
INFO Inbound GNUTella Connect accept	12	0.00%			12	
WEB-IIS scripts-browse	11	0.00%		11		
ICMP Dest ination Unreachable (Protocol Unreachable)	10	0.00%			10	
Possible troj an server activity	10	0.00%		6	4	
SCAN FIN	7	0.00%		7		
INFO Napster C lient Data	6	0.00%			6	
WEB-CGI sc riptalias access	6	0.00%		6		
ICMP Echo Request Broadscan Smurf Scanner	5	0.00%				5
SUNRPC highport access!	5	0.00%		4		1
Back Orifice	5	0.00%		1		4
WEB-MISC compaq nsight directory tra versal	4	0.00%		4		
Attem pted Sun RPC hig h port access	4	0.00%				4
Virus - Possible scr Worm	4	0.00%		4		
WEB-CGI form mail access	4	0.00%		4		
RPC tc p traff ic contains bin_sh	4	0.00%		4		
Port 55850 udp - Possible myserver activity - ref. 010313-1	3	0.00%		1		2
MISC traceroute	3	0.00%		3		
TFTP - External UDP connection to internal tftp server	2	0.00%		2		
Virus - Possible pif W orm	2	0.00%		2		
WEB-CGI redirect access	2	0.00%		2		
Probable NMAP fingerprin t attem pt	2	0.00%		1		1
ICMP Echo Request Sun Solaris	2	0.00%				2
WEB-MISC /....	1	0.00%		1		
External RPC ca ll	1	0.00%		1		
WEB-MISC ftp attempt	1	0.00%		1		
MISC source port 53 to <1024	1	0.00%		1		
INFO - We b File Copied ok	1	0.00%			1	
Total	**253,685**	**Over 80 Unique Alert Types**				

As there is no information on the internal hosts and network architecture, by looking at the scan and alert patterns like source and destination of "TCP *(UDP as well)* SRC and DST Outside Network", and "connect to 515 from inside", I have made the assumption that 130.85.0.0/16 should be the home network a ddress. *The statistic above was generat ed with this assumption in place.*

The table above shows the alerts statistic on each alert type. There are 80 unique alert types during these 5 days. Each alert type has been broken down by the

directions of the traffic, i.e. External Host to External Host, External to Internal, Internal to External and Internal to Internal.

The table above highlights the alerts by descending order of the occurrence of each alert type. However, it does not mean that the most frequent alerts was considered as high severity or high priority. Example is that the most frequent alerts “SMB Name Wildcard” were triggered internally (i.e. from internal to internal), which in normal case will be considered as normal for a network of MS Windows hosts.

To further drill down to the alerts triggered, the table showing the top 15 source and destination pairs are generated.

Source Address	Src Port	Destination Address	Dst Port	Alert	Times
212.179.103.110	1550	MY.NET.88.162	1214	Watchlist 000220 IL-ISDNNET-990517	10246
202.210.163.74	2042	MY.NET.152.20	2401	MISC Large UDP Packet	7865
MY.NET.88.138	1030	MY.NET.150.231	161	SNMP public access	4642
202.210.163.67	3478	MY.NET.152.20	2388	MISC Large UDP Packet	3533
216.22.46.73	25907	MY.NET.153.157	3451	MISC Large UDP Packet	3056
MY.NET.70.177	1106	MY.NET.5.97	161	SNMP public access	2498
MY.NET.70.177	1106	MY.NET.5.127	161	SNMP public access	2494
MY.NET.70.177	1106	MY.NET.5.96	161	SNMP public access	2488
61.96.135.16	3574	MY.NET.88.246	1112	MISC Large UDP Packet	2296
61.96.135.16	3893	MY.NET.88.246	3862	MISC Large UDP Packet	2074
140.142.8.73	1361	MY.NET.153.159	3937	MISC Large UDP Packet	1938
MY.NET.153.220	1031	MY.NET.152.109	161	SNMP public access	1932
MY.NET.88.145	1044	MY.NET.150.195	161	SNMP public access	1728
MY.NET.88.212	1039	MY.NET.150.195	161	SNMP public access	1720
MY.NET.88.155	4040	216.241.219.28	80	spp_http_decode: CGI Null Byte attack detected	1626

As some of the alerts in the top of the list may not be considered as high severity, I have tried to analyse the top 5 alerts and top 5 high severity alerts (according to the standard Snort rules).

Top 5 Alerts #1 - SMB Name Wildcard

Alert Overview

The Server Message Block Protocol (SMB protocol) provides a method for client applications in a computer to read and write to files on and to request services from server programs in a computer network. These alerts were triggered by the SMB (137/udp) communication. Microsoft Windows operating systems since Windows 95 include client and server SMB protocol support. For UNIX systems, a shareware program, Samba, is available.

As seen from the Alert Statistic table, there are more than 50K of this alerts triggered and only 28 of them are triggered by external hosts. Most of them are the communication between the internal hosts. The alerts triggered by the external host(s) should be reviewed.

Top 10 Sources Triggering this Alerts

Source Address	Occurrence
MY.NET.11.7	12,310
MY.NET.11.6	9,849
MY.NET.5.89	3,126
MY.NET.11.5	1,113
MY.NET.152.175	1,061
MY.NET.152.160	898
MY.NET.152.180	537
MY.NET.152.251	522
MY.NET.152.166	521
MY.NET.152.178	520

Top 10 Destinations on this Alerts

Destination Address	Occurrence
MY.NET.11.7	12,260
MY.NET.11.6	9,811
MY.NET.11.5	1,075
MY.NET.152.175	1,064
MY.NET.152.160	898
MY.NET.152.180	541
MY.NET.152.178	528
MY.NET.152.251	520
MY.NET.152.166	517
MY.NET.5.4	510

Top 10 Pairs on this Alerts

Source Address	Destination Address	Occurrence
MY.NET.152.175	MY.NET.11.6	759
MY.NET.11.6	MY.NET.152.175	759
MY.NET.152.160	MY.NET.11.7	492
MY.NET.11.7	MY.NET.152.160	487
MY.NET.11.7	MY.NET.152.178	485
MY.NET.152.178	MY.NET.11.7	475
MY.NET.11.6	MY.NET.152.167	470
MY.NET.152.19	MY.NET.11.7	466
MY.NET.11.6	MY.NET.152.20	465
MY.NET.11.7	MY.NET.152.19	463

Although the top 10 pairs are all internal hosts, which are usually normal within the home network, I have highlighted the Source addresses of the external hosts below:

Source Address	Destination Address	Occurrence
216.150.152.145	MY.NET.5.45	22
216.150.152.141	MY.NET.5.45	6

It is recommended that the host MY.NET.5.45 should have the system administrator to check if the host has already compromised. Here is the 'whois' result from ARIN.

```
Search results for: ! NET-216-150-150-0-1

OrgName:    Easy CGI
OrgID:      EASYCG

NetRange:   216.150.150.0  - 216.150.157.255
CIDR:       216.150.150.0/23, 216.150.152.0/22, 216.150.156.0/23
NetName:    EASYCGI-150-157
NetHandle:  NET-216-150-150-0-1
Parent:     NET-216-150-128-0-1
NetType:    Reassigned
NameServer: NS1.EASY-CGI.COM
NameServer: NS2.EASY-CGI.COM
Comment:
RegDate:     2002-06-19
Updated:    2002-08-08

TechHandle: DR1363-ARIN
TechName:   Richfield, Dan
TechPhone:  +1-845-348-7698
TechEmail:  mail@easycgi.com
```

Correlation

Similar alert has been reported by previous GCIA student Steven Drew (GCIA ID: 0530). That report mentioned that the alerts were triggered by an external host 24.188.117.164 connecting to the internal hosts using “SMB Name Wildcard”.

Defensive Recommendations

1) Filter the SMB (137/udp) from external networks to the home network from the perimeter firewall/router;
2) To further protect the internal hosts using Windows platform, filtering inbound 137-139/udp, 137-139/tcp, and 135/tcp is recommended, as this disallows the Windows shares to be exposed to the public networks and thus reduces the risks of being compromised;

Top 5 Alerts #2 - spp_http_decode: IIS Unicode attack detected

Alert Overview

IIS 4.0 and 5.0 allows remote attackers to read documents outside of the web root, and possibly execute arbitrary commands, via malformed URLs that contain UNICODE encoded characters, a.k.a. the "Web Server Folder Traversal" vulnerability. [http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=IIS+Unicode]

As the Snort rule used to detect this attack is not available, here may be the standard rule that similar to the snort rule that trigger this alert:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB
-IIS Unicode2.pl script (File permission canonicalization)";
uricontent:"/sensepost.exe"; flow:to_server,established; nocase;
classtype:web-application-activity; sid:989;  rev:5;)
```

Top 10 Sources Triggering this Alerts

Source Address	Occurrence
MY.NET.88.246	5,397
MY.NET.153.143	4,878
MY.NET.88.154	2,984
MY.NET.88.201	2,639
MY.NET.153.168	2,383
MY.NET.153.160	1,964
MY.NET.153.211	1,545
MY.NET.153.174	1,476
MY.NET.153.106	1,003
MY.NET.153.196	950

Top 10 Destinations on this Alerts

Destination Address	Occurrence
211.63.185.30	5,719
211.115.213.207	4,943
211.115.213.202	3,018
211.218.200.55	938
203.236.122.7	924
211.233.29.215	755
211.239.123.75	675
211.32.117.26	669
211.239.164.180	641

Top 10 Pairs on this Alerts

Source Address	Destination Address	Occurrence
MY.NET.88.246	211.63.185.30	3,899
MY.NET.153.143	211.115.213.207	3,633
MY.NET.88.154	211.63.185.30	1,808
MY.NET.153.160	203.236.122.7	924
MY.NET.153.143	211.115.213.202	691
MY.NET.153.160	211.115.213.207	678
MY.NET.153.112	211.32.117.26	582
MY.NET.88.201	211.239.164.180	436
MY.NET.88.154	211.115.213.202	416
MY.NET.153.115	211.115.213.202	400

The above table shown that the top ten pairs were quite matched with the top ten sources and top ten destinations. The whois result from APNIC shows that the top

```
inetnum:      211.52.0.0  - 211.63.255.255
inetnum:      211.104.0.0  - 211.119.255.255
inetnum:      203.232.0.0  - 203.239.255.255
netname:      KRNIC-KR
descr:        KRNIC
descr:        Korea Network Information Center
country:      KR
admin-c:      HM127-AP
tech-c:       HM127-AP
remarks:      ****************************************
remarks:      KRNIC is the Na tional Internet Registry
remarks:      in Korea under APNIC. If you would like to
remarks:      find assignment information in detail
remarks:      please refer to the KRNIC Whois DB
remarks:      http://whois.nic.or.kr/english/index.html
remarks:      ****************************************
mnt-by:       APNIC-HM
mnt-lower:    MNT-KRNIC-AP
```

ten destinations are the networks in Korea.

Unicode may be implemented in the remote hosts as the foreign character sets are usually in Unicode, which may triggered the Snort alerts. Depending on the actual traffic from home network to these Korean web sites, the traffic may be considered as abnormal. However, for a campus network, which most likely should have Korean students visiting the Korean web sites, I would consider this as false positive.

Correlation

As this should be a false positive, no correlation is available.

Defensive Recommendations

There is no defensive recommendation to be provided as nearly all of them are outgoing traffic using port 80. For a campus network, it is most likely not possible to deny outgoing traffic using port 80 (i.e. HTTP).

To protect the web servers within the home network, it is recommended to install the latest MS IIS security patches on all public web servers to remove such vulnerabilities.

Top 5 Alerts #3 - SNMP public access

Alert Overview

SNMP is the protocol for monitoring of devices and their functions. SNMP server polls the devices by providing a Community String, and if this string matched, the device will response to the SNMP server according to the poll. Most of the vendors have their factory default set to “Public” as the community string for read only polling, and set to “Private” for read/write configuration from/to the device.

These alerts were triggered by the SNMP protocol communication, with “public” community string unchanged. This will result in leaking of device information such as its uptime, interface I/O, etc.

As the Snort rule used to detect this attack is standard, here is the snort rule that trigger this alert:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 161 (msg:"SNMP public access
 udp"; content:"public"; reference:cve,CAN-2002-0012; reference:cve,CAN-
2002-0013; sid:1411; rev:2; classtype:attempted-recon;)
```

Top 10 Sources Triggering this Alerts

Source Address	Occurrence
MY.NET.70.177	11,698
MY.NET.88.138	5,894

MY.NET.88.225	2,060
MY.NET.153.220	2,015
MY.NET.88.207	1,742
MY.NET.88.136	1,740
MY.NET.88.159	1,736
MY.NET.88.145	1,728
MY.NET.88.181	1,726
MY.NET.88.212	1,720

Top 10 Destinations on this Alerts

Destination Address	Occurrence
MY.NET.150.195	12,278
MY.NET.150.231	5,905
MY.NET.152.109	2,872
MY.NET.5.97	2,498
MY.NET.5.127	2,494
MY.NET.5.96	2,488
MY.NET.150.84	2,070
MY.NET.150.147	1,305
MY.NET.5.95	856
MY.NET.5.92	856

Top 10 Pairs on this Alerts

Source Address	Destination Address	Occurrence
MY.NET.88.138	MY.NET.150.231	5,894
MY.NET.70.177	MY.NET.5.97	2,498
MY.NET.70.177	MY.NET.5.127	2,494
MY.NET.70.177	MY.NET.5.96	2,488
MY.NET.88.225	MY.NET.150.84	2,060
MY.NET.153.220	MY.NET.152.109	2,015
MY.NET.88.207	MY.NET.150.195	1,742
MY.NET.88.136	MY.NET.150.195	1,740
MY.NET.88.159	MY.NET.150.195	1,736
MY.NET.88.145	MY.NET.150.195	1,728

As shown in the Alert Statistic table, all of these alerts are triggered by internal hosts, i.e. no external host involved in this alert. The source MY.NET.70.177 was quite

actively polled the devices in the network segment MY.NET.5.0/24, and the destination MY.NET.150.195 frequently was polled by different hosts. As SNMP should be used mainly for device monitoring, according to the logs, the MY.NET.70.177 may be an SNMP server that monitoring MY.NET.5.0/24. However, the other hosts seem launching the SNMP vulnerability exploitations to the other internal hosts. It is recommended that the system administrator should check out such activities by identify the users during the alerts detected.

Correlation

Details of the SNMP protocol implementation vulnerabilities can be found in http://www.cert.org/advisories/CA-2002-03.html. Vulnerabilities in the decoding and subsequent processing of SNMP messages by both managers and agents may result in denial-of-service conditions, format string vulnerabilities, and buffer overflows. Some vulnerabilities do not require the SNMP message to use the correct SNMP community string.

These vulnerabilities have been assigned the CVE identifiers CAN-2002-0012 and CAN-2002-0013, respectively.

Defensive Recommendations

1) Apply the latest security patches from the vendor to fix the SNMP implementation vulnerabilities;
2) Although some vulnerabilities do not require correct community string, it is highly recommend the system administrator to change all the factor default SNMP string, esp. read/write community, before put the device into production environment;
3) Use filter rules to deny all SNMP requests except from legitimate SNMP monitoring server(s) in the *local network* (as SNMP communicate in plain text transmission so it is highly recommend not being used over public networks);
4) Additionally, it is recommended that the filter rules also include Anti-spoofing mechanism to prevent the spoof of SNMP server address. Device like router should have more than one interface, so system administrator should clearly define which interface the legitimate SNMP server resided.

Top 5 Alerts #4 - MISC Large UDP Packets

Alert Overview

Stateful UDP sessions are normally using small UDP packets, having a payload of not more than 10 bytes. The alerts were triggered because the packets are bigger than 4000 bytes, which is considered as abnormal. This should be either a DoS attack or a convert channel communication between the Trojan hosts and agents. [http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids247&view=research]

As the Snort rule used to detect this attack is standard, here is the snort rule that trigger this alert:

```
misc.rules:alert udp $EXTERNAL_NET any  -> $HOME_NET any (msg:"MISC Large
UDP Packet"; dsize: >4000; reference:arachnids,247; classtype:bad -unknown;
sid:521; rev:1;)
```

Top 10 Sources Triggering this Alerts

Source Address	Occurrence
202.210.163.74	7,889
61.96.135.16	6,568
202.210.163.67	4,123
210.220.160.159	3,484
216.22.46.73	3,060
140.142.8.73	1,942
61.153.17.24	1,624
211.110.11.194	1,370
211.233.82.153	596
211.63.185.21	560

Top 10 Destinations on this Alerts

Destination Address	Occurrence
MY.NET.152.20	12,012
MY.NET.88.246	6,568
MY.NET.153.190	3,484
MY.NET.153.157	3,060
MY.NET.153.159	1,991
MY.NET.152.157	1,624
MY.NET.153.153	1,074
MY.NET.153.126	596
MY.NET.88.154	560
MY.NET.152.159	357

Top 10 Pairs on this Alerts

Source Address	Destination Address	Occurrence
202.210.163.74	MY.NET.152.20	7,889
61.96.135.16	MY.NET.88.246	6,568
202.210.163.67	MY.NET.152.20	4,123
210.220.160.159	MY.NET.153.190	3,484
216.22.46.73	MY.NET.153.157	3,060
140.142.8.73	MY.NET.153.159	1,942

61.153.17.24	MY.NET.152.157	1,624
211.110.11.194	MY.NET.153.153	1,074
211.233.82.153	MY.NET.153.126	596
211.63.185.21	MY.NET.88.154	560

As shown in this table, all of the top 10 pairs are triggered by the external hosts to internal hosts. After further query with source and destination ports, it shown that the source and destination ports of these communications are not focused on certain UDP ports, except some of them are using port 0.

Here are some whois query results to show the information of the source networks:

```
inetnum:      202.208.0.0  - 202.211.255.255
netname:      JPNIC-NET-JP
descr:        Japan Network Information Center
country:      JP
admin-c:      JNIC1-AP
tech-c:       JNIC1-AP
remarks:      JPNIC Allocation Block
```

```
inetnum:      61.96.0.0  - 61.111.255.255
inetnum:      210.220.0.0  - 210.223.255.255
netname:      KRNIC-KR
descr:        KRNIC
descr:        Korea Network Information Center
country:      KR
admin-c:      HM127-AP
tech-c:       HM127-AP
```

```
OrgName:    ServInt Corp.
OrgID:      SRVN

NetRange:   216.22.0.0 - 216.22.47.255
CIDR:       216.22.0.0/19, 216.22.32.0/20
NetName:    SERVINT-CIDR-2
NetHandle:  NET-216-22-0-0-1
Parent:     NET-216-0-0-0-0
NetType:    Direct Allocation
NameServer: NS.SERVINT.COM
NameServer: NS2.SERVINT.COM
```

Most of the source hosts are from Japan and Korea.

Correlation

The alert details can be found in http://www.whitehats.com/info/IDS247. The false positive cases mentioned in the page seem not matched with the alerts detected as the destination ports or packet size mentioned are not matched.

Defensive Recommendations

1) Implement the filter rules that deny all incoming traffic unless explicitly allowed;
2) Setup the host based IDS in critical servers to drop all suspicious traffic

Top 5 Alerts #5 - ICMP Echo Request L3retriever Ping

Alert Overview

Retriever is a proactive network security management tool that automatically discovers and maps network components, unobtrusively identifies vulnerabilities, provides safeguard and policy recommendations, and performs customizable network audits. [http://www.symantec.com/symadvantage/005/newprod.html]

As the Snort rule used to detect this attack is standard, here is the snort rule that trigger this alert:

```
icmp.rules:alert icmp $EXTERNAL_NET any  -> $HOME_NET any (msg:"ICMP
L3retriever Ping"; content: "ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI"; itype: 8;
icode: 0; depth: 32; reference:arachnids,311; classtype:attempted-recon;
sid:466; rev:1;)
```

This signature is based on the characteristic ping of the L3 Networks security scanner called "Retriever 1.5". These probes should be rare, since the software is usually restricted to limited IP address ranges. This may also be a false positive as this type of ICMP ping seems to be also generated by (plain) Win2K host talking to Win2K domain controllers.
[http://www.whitehats.com/info/IDS311]

Top 10 Sources Triggering this Alerts

Source Address	Occurrence
MY.NET.152.175	1,068
MY.NET.152.160	914
MY.NET.152.180	548
MY.NET.152.178	532
MY.NET.152.166	523
MY.NET.152.172	514
MY.NET.152.251	513
MY.NET.152.21	512
MY.NET.152.245	498
MY.NET.152.250	497

ALL Destinations on this Alerts

Destination Address	Occurrence
MY.NET.11.7	12,314

MY.NET.11.6	9,919
MY.NET.11.5	1,110
MY.NET.5.4	331
MY.NET.150.197	44
MY.NET.5.45	27
MY.NET.153.220	16
MY.NET.5.35	11
MY.NET.5.7	5
MY.NET.130.187	4
MY.NET.5.3	2
MY.NET.5.96	1

Top 10 Pairs on this Alerts

Source Address	Destination Address	Occurrence
MY.NET.152.175	MY.NET.11.6	770
MY.NET.152.160	MY.NET.11.7	496
MY.NET.152.178	MY.NET.11.7	485
MY.NET.152.247	MY.NET.11.6	472
MY.NET.152.167	MY.NET.11.6	470
MY.NET.152.163	MY.NET.11.6	465
MY.NET.152.157	MY.NET.11.6	463
MY.NET.152.169	MY.NET.11.7	463
MY.NET.152.179	MY.NET.11.7	461
MY.NET.152.171	MY.NET.11.7	457

As mentioned in the above, the L3 security tool should be restricted to limited IP but from the alert logs, there are 96 unique sources "pinging" the destinations, which most are destined to MY.NET.11.6 and MY.NET.11.7. It is obvious the false positive case mentioned, i.e. MY.NET.11.6 and MY.NET.11.7 are Windows 2000 Domain Controllers and the sources in the alerts are Windows 2000 hosts.

Although the top 10 pairs are all internal hosts, I have highlighted the Source addresses of the external hosts below:

Source Address	Destination Address	Occurrence
216.150.152.145	MY.NET.5.45	16
216.150.152.141	MY.NET.5.45	11

These alerts seems not to be false positives because the source addresses are specific to 2 external hosts. As analysed in the alert "SNMP public access", MY.NET.5.0/24 may be a server segment because there are lots of SNMP traffic destined to this segment. These 2 source hosts may performing reconnaissance

exercise to the MY.NET.5.45. It is recommended that the system administration should further review any traffic coming from these 2 hosts. Here are the related alerts generated by the source 216.150.152.0/24:

Source Address	Description	Occurrence
216.150.152.145	SMB Name Wildcard	22
216.150.152.145	ICMP Echo Request Windows	21
216.150.152.145	ICMP Echo Request L3retriever Ping	16
216.150.152.141	ICMP Echo Request L3retriever Ping	11
216.150.152.141	SMB Name Wildcard	6

Also note that all these traffics are destined to MY.NET.5.45 as well.

Correlation

This alert is mentioned in
http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids311&view=event.

Defensive Recommendations

1) Implement filtering rule to deny all incoming ICMP from external networks. If PING will be used for troubleshooting, the rule should allow only incoming Echo Reply and outgoing Echo Request;

Top 5 High Severity Alerts #1 - Watchlist 000220 IL-ISDNNET-990517

Alert Overview

With reference to GCIA practical paper, netname IL-ISDNET-990517 is the ISDN Net Ltd (212.179.0.0/17), which is well-known attack sites. Suspicious traffic between the internal network and this network should be further analysed.

Top 10 Sources Triggering this Alerts

Source Address	Occurrence
212.179.103.110	10,328
212.179.35.118	486
212.179.38.171	44
212.179.35.6	37
212.179.66.17	36
212.179.126.3	28
212.179.97.86	24
212.179.98.71	22
212.179.27.111	9
212.179.75.135	9

ALL Destinations on this Alerts

Destination Address	Occurrence
MY.NET.88.162	10,505
MY.NET.153.190	312
MY.NET.152.159	224
MY.NET.153.164	37
MY.NET.150.133	20
MY.NET.150.220	7
MY.NET.153.191	5
MY.NET.152.247	1
MY.NET.153.197	1

Top 10 Pairs on this Alerts

As these alerts are triggered by the source address, it is necessary to also find out which destination ports they are tried to connect to.

Source	Src Port	Destination	Dst Port	Occurrence
212.179.103.110	1550	MY.NET.88.162	1214	10,246
212.179.35.118	80	MY.NET.153.190	1478	262
212.179.35.118	80	MY.NET.152.159	4807	224
212.179.38.171	1121	MY.NET.88.162	1214	44
212.179.98.71	1048	MY.NET.88.162	1214	22
212.179.126.3	38796	MY.NET.88.162	1214	18
212.179.35.6	80	MY.NET.153.164	1741	16
212.179.66.17	80	MY.NET.153.190	1500	12
212.179.35.6	80	MY.NET.153.164	1721	11
212.179.35.6	80	MY.NET.153.164	2422	10

Correlation

The destination port is 1214, which is the top 1 destination port in the scan log. (Please refer to the "Analysis of Scans Data" in next section for details)

Defensive Recommendations

1) Implement the filter rules that deny all incoming traffic unless explicitly allowed;
2) Setup the host based IDS in critical servers to drop all suspicious traffic

Top 5 High Severity Alerts #2 - spp_http_decode: CGI Null Byte attack detected

Alert Overview

Remote attackers try to view the source code for CGI programs via a null character (%00) at the end of a URL. Such CGI programs may have sensitive information such as hard-coded password, database user ID and password. This allows the remote attackers to gain more information for further exploitation.

As there are not too many hosts involved, I have listed only their communication pairs without showing the Top 10 Sources and Top 10 Destinations.

ALL Pairs on this Alerts

Source	Destination	Occurrence
MY.NET.88.155	216.241.219.28	7,251
MY.NET.153.171	209.10.239.135	264

MY.NET.152.19	206.128.186.13	60
MY.NET.153.163	205.188.180.25	4
MY.NET.152.182	131.118.254.40	2
MY.NET.150.103	64.4.36.250	1
MY.NET.152.13	205.188.180.25	1
MY.NET.153.202	128.167.120.48	1
MY.NET.88.216	MY.NET.11.4	1

Correlation

The alert, which similar to this one, has been described in the CVE-2000-0149.

Defensive Recommendations

1) Update the security patches to remove this vulnerability;
2) Deny all incoming HTTP traffic unless explicitly allowed, such as to web servers.

Top 5 High Severity Alerts #3 - WEB-MISC Attempt to execute cmd

Alert Overview

These alerts are triggered by sending a HTTP GET request with cmd.exe within the URL. Most likely, these should be related to Nimda, CodeRed II and sadminIIS worm. The infected host (usually IIS web server) has its /winnt/system32/cmd.exe copied into /scripts of the web server document root directory, which is executable on IIS web server using HTTP.

These are the Snort rules that may trigger the alert:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS
cmd32.exe access"; flow:to_server,established; content:"cmd32.exe"; nocase;
classtype:web-application-attack; sid:1661;  rev:3;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg: "WEB-IIS
cmd.exe access"; flow:to_server,established; content:"cmd.exe"; nocase;
classtype:web-application-attack; sid:1002;  rev:5;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS cmd?
access";flow:to_server,established; content:" .cmd?&"; nocase;
classtype:web-application-attack; sid:1003;  rev:6;)
```

Top 10 Sources Triggering this Alerts

Source Address	Occurrence
80.24.32.98	430
24.112.145.222	359
213.46.104.229	301

129.49.201.161	196
194.20.229.34	122
24.138.61.171	97
211.91.20.135	95
80.143.249.243	80
68.39.188.16	74
24.84.115.28	57

Top 10 Destinations on this Alerts

Destination Address	Occurrence
MY.NET.150.198	339
MY.NET.5.96	210
MY.NET.5.92	192
MY.NET.5.95	190
MY.NET.5.14	184
MY.NET.150.101	114
MY.NET.150.228	108
MY.NET.150.143	107
MY.NET.150.246	104
MY.NET.150.59	102

Top 10 pairs on this Alerts

Source	Destination	Occurrence
213.46.104.229	MY.NET.5.95	80
213.46.104.229	MY.NET.5.92	78
68.39.188.16	MY.NET.150.198	74
213.46.104.229	MY.NET.5.96	73
213.46.104.229	MY.NET.5.14	70
80.24.32.98	MY.NET.5.95	53
80.24.32.98	MY.NET.5.96	51
80.24.32.98	MY.NET.150.228	50
80.24.32.98	MY.NET.150.143	49
80.24.32.98	MY.NET.5.14	47

Most of the alerts are triggered by external hosts. From the experience gained, most of them should be false positive, and not all hosts are Windows platform with IIS installed. Unless the internal hosts' information obtained, it is difficult to conclude which of them have virus infected.

Correlation

This alert is quite common. There are lots of references available in security related web site. Here are some of them:

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms00-078.asp

http:/www.cert.org/advisories/CA-2001-26.html

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-018.asp

Defensive Recommendations

1) Apply the cumulative patches on IIS web server, which can be obtained in http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-018.asp
2) Uninstall/Stop IIS services immediately unless absolutely necessary;
3) Install Anti-virus software in all the machines, clean the related viruses/worms immediately;
4) Implement filtering rule that deny all incoming HTTP traffic from outside network, unless explicitly allowed;

Top 5 High Severity Alerts #4 - FTP DoS ftpd globbing

Alert Overview

Many FTP servers also implement globbing, so that the command mget *.c means retrieve all the files ending in ".c," and get ~foo/file.name means get the file named "file.name" in the home directory of foo.

For this vulnerability, intruders can execute arbitrary code with the permissions of the process running the FTP server, or cause a DoS on the FTP server.

[http://www.cert.org/advisories/CA-2001-07.html]

Top 10 Sources Triggering this Alerts

Source Address	Occurrence
12.108.116.50	465
169.232.107.98	410
217.110.4.98	172
200.177.253.2	153
169.232.73.61	148
68.46.105.227	131
192.35.232.13	122
208.47.17.5	95

64.122.18.17	95
198.209.32.253	89

All Destinations on this Alerts

Destination Address	Occurrence
MY.NET.153.191	1,824
MY.NET.153.187	148
MY.NET.153.216	110
MY.NET.153.190	97
MY.NET.153.165	24

Obviously, these hosts are running FTP daemon.

Top 10 Pairs on this Alerts

Source	Destination	Occurrence
12.108.116.50	MY.NET.153.191	465
169.232.107.98	MY.NET.153.191	410
217.110.4.98	MY.NET.153.191	172
200.177.253.2	MY.NET.153.191	153
169.232.73.61	MY.NET.153.187	148
68.46.105.227	MY.NET.153.191	131
192.35.232.13	MY.NET.153.191	122
208.47.17.5	MY.NET.153.191	95
64.122.18.17	MY.NET.153.191	95
198.209.32.253	MY.NET.153.191	89

All Top 10 source hosts are from external networks and most of them are connected to MY.NET.153.191. It is recommended that the system administrator should further check these internal FTP hosts are running vulnerable FTP daemon. Details can refer to http://www.cert.org/advisories/CA-2001-07.html which contains the vulnerable list by vendor and version.

Correlation

There are several related CVE and Candidates entries related to this alert (CAN-2001-0247, CAN-2001-0248, CAN-2001-0249 and CAN-2001-0421). These are related to the buffer overflow in FTP server allows remote attackers to execute arbitrary commands.

http://www.cert.org/advisories/CA-2001-07.html

Defensive Recommendations

1) Apply the patches on FTP server where details can be found in CERT advisories page;
2) Uninstall/Stop FTP services immediately unless absolutely necessary;
3) Implement filtering rule that deny all incoming FTP traffic from outside network, unless explicitly allowed;

Top 5 High Severity Alerts #5 - Watchlist 000222 NET-NCFC

Alert Overview

Similar to Watchlist 000220, NET-NCFC is the Institute of Computing Technology Chinese Academy of Sciences, Beijing, China (159.226.0.0/16).

The Pair on this Alerts

As these alerts are triggered by the source address, it is necessary to also find out which destination ports they are tried to connect to.

Source	Src Port	Destination	Dst Port	Occurrence
159.226.2.10	80	MY.NET.88.140	2543	38
159.226.2.10	80	MY.NET.88.140	2546	16
159.226.2.10	80	MY.NET.88.140	2548	15
159.226.2.10	80	MY.NET.88.140	2545	13
159.226.2.10	80	MY.NET.88.140	2573	12
159.226.2.10	80	MY.NET.88.140	2571	11
159.226.2.10	80	MY.NET.88.140	2547	9
159.226.2.10	80	MY.NET.88.140	2574	7
159.226.2.10	80	MY.NET.88.140	2575	5
159.226.2.10	80	MY.NET.88.140	2555	4

All alerts are triggered by only these 2 hosts (159.226.2.10 and MY.NET.88.140). According to this alert pattern, the external host seems performing reconnaissance exercise to determine which port(s) the host is listening. After further query, there are no other alerts triggered by this source.

Defensive Recommendations

1) Implement filtering rule to deny all incoming traffic from external networks, unless explicitly allowed.

Analysis of Scans Data

Top 15 Scanning Hosts' Addresses

Source	Occurrence	%
MY.NET.88.162	1,930,341	46.25%
MY.NET.5.89	612,435	14.67%
MY.NET.60.43	359,536	8.61%
MY.NET.11.8	149,686	3.59%
MY.NET.153.191	124,318	2.98%
MY.NET.253.10	63,987	1.53%
MY.NET.6.49	35,609	0.85%
MY.NET.6.45	34,653	0.83%
MY.NET.6.50	34,605	0.83%
MY.NET.153.189	33,718	0.81%
MY.NET.153.190	32,620	0.78%
MY.NET.6.51	28,183	0.68%
MY.NET.6.52	26,840	0.64%
12.151.57.38	23,615	0.57%
MY.NET.6.60	19,534	0.47%
Total	4,174,129	84.08%

From the query result, there are 591 unique source addresses were detected including both internal and external hosts. As shown in the above diagram, MY.NET.88.162 and MY.NET.5.89 have triggered comparatively large number of alerts. More than 46% of the scanning alerts are triggered by MY.NET.88.162. As the top 7 – 15 source addresses seem to be active, so I have listed out top 15 instead of top 10 source addresses.

Top 10 Destination Hosts' Addresses

Destination	Occurrence	%
MY.NET.1.3	49,548	1.19%
MY.NET.153.157	35,990	0.86%
MY.NET.150.198	32,002	0.77%
MY.NET.1.4	27,052	0.65%
MY.NET.11.7	25,999	0.62%
MY.NET.88.245	23,625	0.57%
MY.NET.6.45	21,997	0.53%
MY.NET.11.6	21,801	0.52%
MY.NET.60.43	20,030	0.48%
MY.NET.150.197	18,837	0.45%

Total	4,174,129	6.63%

From the query result, 220,540 unique destination addresses were detected including both internal and external hosts. The percentage among the top 10 seems quite even, and there is no outstanding host being spotted.

Top Scanning Destination Ports

Destination Port	Occurrences	Description
1214/udp	2,024,614	KaZaA or Morpheus - MP3 Sharing program
161/udp	610,925	SNMP
80/tcp	178,469	HTTP
1346/udp	149,936	Alta Analytics License Manager
7001/udp	117,784	afs3-callback callbacks to cache managers; or Freak88 Trojan
7000/udp	82,216	afs3-fileserver; or Remote Grab, Kazimas Trojan
53/udp	78,834	DNS
137/udp	45,879	NETBIOS Name Service
0/udp	45,205	Reserved
6970/udp	32,488	GateCrasher Trojan
6346/tcp	31,964	Napster- MP3 Sharing program
7003/udp	21,585	afs3-vlserver - volume location database
1214/tcp	19,136	KaZaA or Morpheus - MP3 Sharing program
139/tcp	13,349	NETBIOS Session Service

From the query result, there are 93,468 unique TCP and UDP destination ports have been scanned among the 220,540 unique destination hosts. 1214/udp, a port for MP3 Sharing program, seems to be heavily used within the home network. 161/udp, a SNMP polling, is the next high occurrence. This may due to the high frequency or large no. of nodes to be polled from the SNMP server. 0/udp scanning seems to be reconnaissance exercise from attackers as port 0 of either TCP or UDP are usually considered abnormal. Besides these, port 6970/tcp, 7001/udp and 7000/udp maybe used by the Trojan programs like SubSever, Freak99, etc. Further investigation on the hosts that listening to these ports are recommended.

Top Scanning Source Ports

Source Port	Occurrences	Description
1214/udp	2,063,324	KaZaA or Morpheus - MP3 Sharing program
1111/udp	609,798	LM Social Server
123/udp	319,863	NTP
1347/udp	149,736	Multi-media Conferencing

7000/udp	117,583	afs3-fileserver; or Remote Grab, Kazimas, SubSeven Trojans
7001/udp	103,800	afs3-callback - Callbacks to Cache Managers; or Freak88 Trojan
0/udp	53,584	Reserved
137/udp	45,981	NETBIOS Name Service
62025/tcp	28,196	UNKNOWN
6970/udp	22,523	GeteCrasher Trojan
49192/tcp	18,251	UNKNOWN
34372/tcp	16,862	UNKNOWN
88/udp	14,548	Kerberos
516/udp	8,552	Videotex

From the query result, there are 49,428 unique TCP and UDP source ports used to scan the network. Again, the 1214/udp seems to be heavily used within the home network. 7000/udp, 7001/udp and 6970/udp should be investigated as this may be the communication between the Trojan agents and hosts.

To further elaborate the above data, further queries have been performed to understand the unusual traffic pattern.

High volume of scan alerts triggered by MY.NET.88.162

Source	Src Port	Destination	Dst Port	Occurrences
MY.NET.88.162	1214/udp	212.179.35.118	1214/udp	3,871
MY.NET.88.162	1214/udp	130.203.203.57	1214/udp	1,475
MY.NET.88.162	1214/udp	24.166.48.174	1214/udp	1,205
MY.NET.88.162	1214/udp	212.107.42.1	1214/udp	1,205
MY.NET.88.162	1214/udp	24.128.103.41	1214/udp	1,196
MY.NET.88.162	1214/udp	143.107.180.5	1214/udp	1,173
MY.NET.88.162	1214/udp	130.91.154.6	1214/udp	880
MY.NET.88.162	1214/udp	24.165.100.198	1214/udp	795

The source MY.NET.88.162 has performed scanning to different destination hosts and ports where most of them are non-home networks. More than 70% of the destination ports scanned by this host is 1214/udp. This host seems to be a MP3 host that shared to public using KaZaA or similar MP3 sharing program. Unless this is intended to install by the user, this host may have been compromised and served as a public MP3 "server". **Further investigation on this host is absolutely needed.**

High volume of scan alerts triggered by MY.NET.5.89

The source MY.NET.5.89 has performed scanning to different hosts with different patterns, so the query result cannot be easily summarized by a small table in this

report. This host tried to use the source port 1111/udp to connect to the port 161/udp of the hosts in the home network. The source port should had been crafted, which can be easily done by using scanning tools like Nmap or Hping2. Also, it also tried to use source port 137/udp to connect to the port 137/udp of the hosts in the home network. Besides, according to the incremental source ports of this hosts, the host tried to repeatedly query the host MY.NET.1.3 using 53/udp more than 1500 times. This also results in MY.NET.1.3 becoming the top host being scanned.

Here are the tiny portion of the scan patterns:

Source	Src Port	Destination	Dst Port
MY.NET.5.89	3090/udp	MY.NET.1.3	53/udp
MY.NET.5.89	3091/udp	MY.NET.1.3	53/udp
MY.NET.5.89	3092/udp	MY.NET.1.3	53/udp
MY.NET.5.89	3093/udp	MY.NET.1.3	53/udp
MY.NET.5.89	3101/udp	MY.NET.1.3	53/udp
MY.NET.5.89	3103/udp	MY.NET.1.3	53/udp
MY.NET.5.89	3104/udp	MY.NET.1.3	53/udp
MY.NET.5.89	3107/udp	MY.NET.1.3	53/udp
MY.NET.5.89	3108/udp	MY.NET.1.3	53/udp
MY.NET.5.89	3109/udp	MY.NET.1.3	53/udp

If the host is not a SNMP server like HP Openview, the scanning of the 161/udp seems abnormal. This scanning should be targeted to find out the vulnerability of the SNMP protocol implementation within the internal network. Such vulnerability cause a denial of service or gain privileges via SNMPv1 trap handling, as demonstrated by the PROTOS c06-SNMPv1 test suite. Related information can be associated with CVE candidate entry CAN-2002-0012. The host may also make use of 137/udp to find out the information of windows hosts within the home network.

From the scanning pattern, the host seems launching attacks either by the user, or already compromised by the attackers for further intrusions. **Further investigation on this host is absolutely needed.**

High volume of scan alerts triggered by MY.NET.60.43

Source	Src Port	Destination	Dst Port	Occurrences
MY.NET.60.43	7000/udp	MY.NET.152.172	7001/udp	3,279
MY.NET.60.43	7000/udp	MY.NET.152.175	7001/udp	1,367
MY.NET.60.43	7000/udp	MY.NET.149.36	7001/udp	1,162
MY.NET.60.43	7000/udp	MY.NET.152.244	7001/udp	1,013
MY.NET.60.43	7000/udp	MY.NET.153.209	7001/udp	1,012
MY.NET.60.43	7000/udp	MY.NET.152.19	7001/udp	959

There are quite a lot of communication between 7000 and 7001 from MY.NET.60.43 to other hosts in the home network. Besides, the host also scanned the MY.NET.153.1xx and MY.NET.153.20x using source port 123/udp to destination ports from 1000 – 5000. The source port seems to be crafted, which can be easily done by using scanning tools like Nmap or Hping2. **Further investigation on this host is absolutely needed.**

Analysis of OOS Data

As the OOS log files analysed did not contain huge amount of entries, some simple scripts/UNIX commands to extract the distinct source and destination IP address have been used.

By just extracting the first line of each OOS alert, the following Top 5 Source and Destination hosts have been listed as below.

Source	Occurrence
211.110.13.28	19
24.226.42.77	9
66.125.92.204	7
64.4.124.151	7
68.47.36.112	5

Destination	Occurence
MY.NET.153.150	9
MY.NET.88.165	7
MY.NET.88.162	7
MY.NET.153.189	6
MY.NET.5.96	4

Top Source Host #1 (211.110.13.28)

The host 211.110.13.28 sent packets to 19 hosts in MY.NET.5.0 within 4 sec (during 02-Jun 13:48:39 to 13:48:43) using both source port and destination port 21. The flags **SF**** have been set in these 19 packets.

The SYN-FIN packet tells the computer to begin a connection and to tear down the connection at the same time. Most attackers uses this technique for network mapping.

Besides, the source port of the packets is 21, which means the packets may have been crafted.

The table below shows the scan alerts from the scan log analysed, which matches the OOS alerts.

Month	Day	Hr	Min	Sec	Source Addr	Src Port	Destination Addr	Dest Port	Protocol	Flag
Jun	2	13	46	41	211.110.13.28	21	MY.NET.5.14	21	SYNFIN	******SF
Jun	2	13	46	41	211.110.13.28	21	MY.NET.5.25	21	SYNFIN	******SF
Jun	2	13	46	42	211.110.13.28	21	MY.NET.5.83	21	SYNFIN	******SF
Jun	2	13	46	42	211.110.13.28	21	MY.NET.5.87	21	SYNFIN	******SF
Jun	2	13	46	42	211.110.13.28	21	MY.NET.5.89	21	SYNFIN	******SF
Jun	2	13	46	42	211.110.13.28	21	MY.NET.5.90	21	SYNFIN	******SF
Jun	2	13	46	42	211.110.13.28	21	MY.NET.5.95	21	SYNFIN	******SF
Jun	2	13	46	42	211.110.13.28	21	MY.NET.5.103	21	SYNFIN	******SF
Jun	2	13	46	42	211.110.13.28	21	MY.NET.5.106	21	SYNFIN	******SF
Jun	2	13	46	42	211.110.13.28	21	MY.NET.5.108	21	SYNFIN	******SF
Jun	2	13	46	42	211.110.13.28	21	MY.NET.5.109	21	SYNFIN	******SF
Jun	2	13	46	42	211.110.13.28	21	MY.NET.5.110	21	SYNFIN	******SF
Jun	2	13	46	43	211.110.13.28	21	MY.NET.5.127	21	SYNFIN	******SF
Jun	2	13	46	43	211.110.13.28	21	MY.NET.5.128	21	SYNFIN	******SF
Jun	2	13	46	43	211.110.13.28	21	MY.NET.5.137	21	SYNFIN	******SF
Jun	2	13	46	43	211.110.13.28	21	MY.NET.5.141	21	SYNFIN	******SF
Jun	2	13	46	43	211.110.13.28	21	MY.NET.5.142	21	SYNFIN	******SF
Jun	2	13	46	43	211.110.13.28	21	MY.NET.5.143	21	SYNFIN	******SF
Jun	2	13	46	45	211.110.13.28	21	MY.NET.5.249	21	SYNFIN	******SF

Top Source Host #2 (24.226.42.77)

The host 24.226.42.77 sent totally 9 packets to MY.NET.153.150 (the Top 1 destination host) port 2331 and 6346 using source port 0 and 2331 respectively. Different combination of flags (such as 2*SF**AU, 21SF*PA*) have been set in these 9 packets.

The first 2 bits (the flags '2' and '1' in snort alert) are reserved flags and should not be set under any conditions. However, some programs such as Hping2 enable an attackers to set these types of bits.

However, there is only 1 corresponding attack alert in the Alert Logs:

Date/Time	Description	Source Addr	Src Port	Destination Addr	Dest Port
Jun-4 18:56: 45.98378	INFO Inbound GNUTella Connect request	24.226.42.77	2331	MY.NET.153.150	6346

Top Source Host #3 (66.125.92.204)

The host 66.125.92.204 sent totally 7 packets to MY.NET.88.162 with DF (Don't Fragment) and MF (More Fragments) flags set. These 2 flags should be not set at the same time under any conditions.

Moreover, the FRAG OFFSET is set to 0x0 so it will overwrite the existing data and put in the first byte of the data field. This is obviously not a reasonable offset value.

The packets should be crafted and since the packets are "fragmented", there are no protocol information in the packets (except they are TCP packets).

Top Source Host #4 (64.4.124.151)

The host 64.4.124.141, similar to the Source Host 24.226.42.77, sent totally 7 packets to MY.NET.88.165 with different combination of flags set. However, the events happened in Jun-13 and since the OOS log files analysed do not include that date, there are no alerts/scan alert information available.

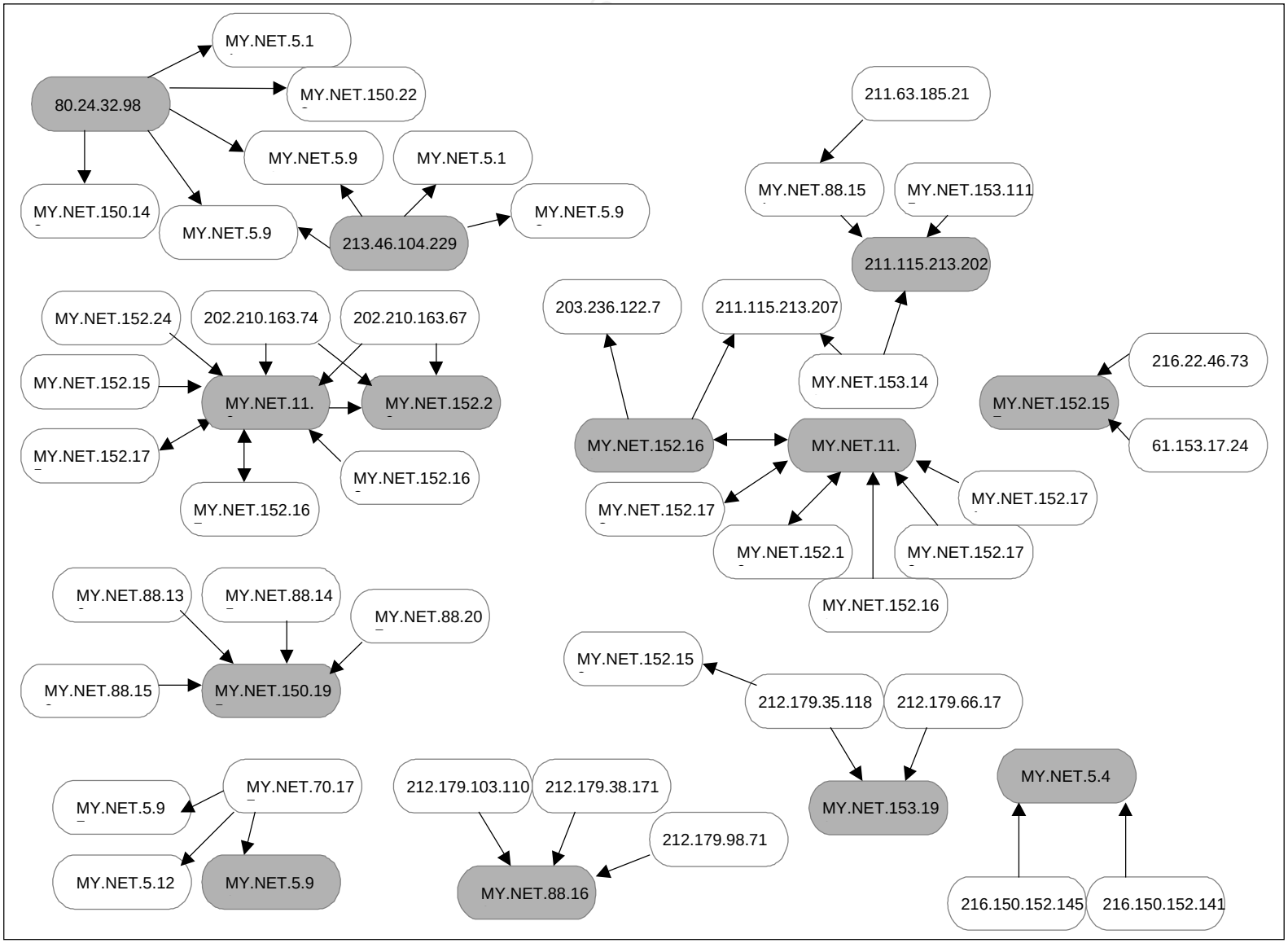
Top Source Host #5 (68.47.36.112)

The host 68.47.36.112 sent totally 5 packets to MY.NET.153.189 port 1323 and using source port 6346.

The table below shows the attack alerts from the attack log analysed, which matches the OOS alerts.

Date/Time	Description	Source Addr	Src Port	Destination Addr	Dest Port
Jun-02 16:56: 25.412556	Null scan!	68.47.36.112	6346	MY.NET.153.189	1323
Jun-02 16:56: 25.412556	Null scan!	68.47.36.112	6346	MY.NET.153.189	1323

Link Graphs



Description of Analysis Processes

First of all, each type of files are concatenated into a single large file of each type. After that, they are imported into MS SQL database further for analysis. Basically, it is necessary to edit the log files in batch mode so that they are formatted into a space-delimiter (or other type of delimiter) files. These can be done by using 'cat', 'sed' and 'grep' in my Linux box.

After that, they are transferred into my Windows platform PC for importing into the MS SQL database. The log files are transferred using removable hard disk so the log integrity are ensured.

Here are some of the simple SQL statement to give me an rough idea about what was going on these 5 days:

To generate the list of alerts detected:

```
Select description, count(*) as occurrence from alert where
description in (select distinct description from alert) group by
description order by occurrence desc
```

To create a view and then find out the occurrence of each alert that involves external hosts as Source Address and Destination Address:

```
/*
Create View ExtExtAlerts as
Select src, srcport, dst, dstport, description, count(*) as
occurrence
from alert2
where description in
(select distinct description from alert2)
and src not like '%MY.NET%' and dst not like '%MY.NET%'
group by src, srcport, dst, dstport, description
*/

/* Select * from ExtExtAlerts */


Select distinct description, sum(occurrence) as TotalSum from
ExtExtAlerts group by description
```

Similar SQL statements have been used to find out the occurrence of each alert that involves:

1) external hosts to internal hosts
2) internal hosts to internal hosts
3) internal hosts to external hosts

To find out the top source and destination addresses of interested attack:

```
/* Top pairs of this attack */
/*
Select src, dst, description, count(*) as occurrence
from alert2
where description like '%ICMP Echo Request L3retriever Ping%'
and src not like '%MY.NET%'
group by src, dst, description
order by occurrence desc
*/
/* Top Source of this attack */

Select src, description, count(*)  as occurrence
from alert2
where description like '%ICMP Echo Request L3retriever Ping%'
group by src, description
order by occurrence desc

/* Top Dest of this attack */
/*
Select dst, description, count(*) as occurrence
from alert2
where description like '%ICMP Echo Request L3retriever Ping%'
group by dst, description
order by occurrence desc
*/
```

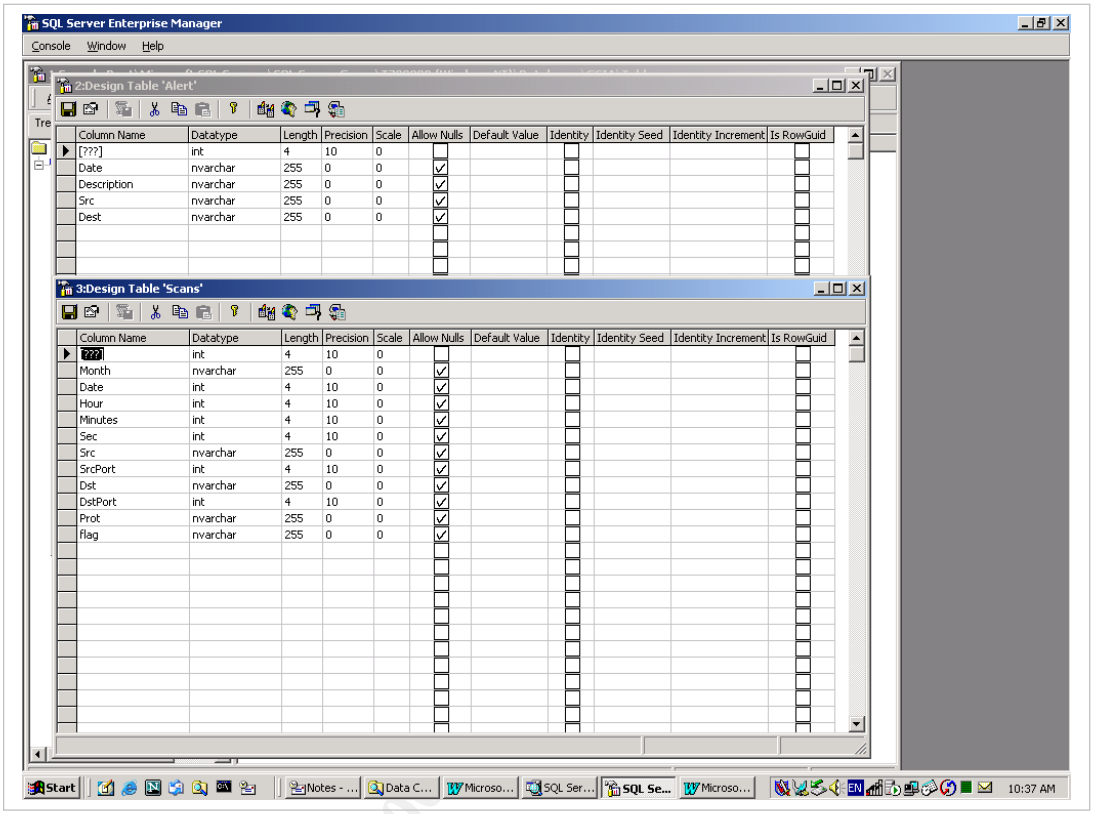
By adding different criteria in the SQL statement (such as the criteria highlighted in **BOLD**), we can also find out the no. of occurrence of the “incoming” or “outgoing” traffic.

To find out the top walkers of Scanning/Scanned hosts:

```
/* Select Top Scanning Host
select src, count(*) as occurrence from scans
where src in
(select distinct src from scans)
group by src
order by occurrence desc
*/

/* Select Top Scanned Host
select dst, count(*) as occurrence from scans
where dst in
(select distinct dst from scans)
group by dst
order by occurrence desc
*/

/* Select Top Src Port */
Select srcport, prot, count(*) as occurrence from scans
where srcport in
(select distinct srcport from scans)
group by srcport, prot
order by occurrence desc


/* Select Top Dst Port
Select dstport, prot, count(*) as occurrence from scans
where dstport in
(select distinct dstport from scans)
group by dstport, prot
order by occurrence desc
*/

/* Select Top Src/Dst Pair
select src, dst, count(*) as occurrence from temp_scans
group by src, dst
order by occurrence desc
*/


/* Select Occurrence of Source & Destination pairs with  Src/Dst
Ports"
select src, srcport, dst, dstport, count(*) as occurrence from
scans
where src in
(select distinct src from scans)
group by src, srcport, dst, dstport
order by occurrence desc
*/
```

Additional Information

Here are the tables' structure of the logs database:



References

"TCP/UDP Port and Service Search" URL: http://www.securitystats.com/tools/portsearch.asp (Sept 14, 2002)

"Trojan Horse Port List" URL: http://www.sys-security.com/html/papers/trojan_list.html (Sept 10, 2002)

"CVE" URL: http://cve.mitre.org/cve (13 Aug, 2002)

"Server Message Block Protocol" URL: http://searchsystemsmanagement.techtarget.com/sDefinition/0,,sid20_gci214214,00.html (Nov 11, 2002)

"APNIC Whois Database" URL: http://www.apnic.net/apnic-bin/whois.pl (15 Aug, 2002)

"CERT Advisory" URL: http://www.cert.org/advisories/CA-2002-03.html (10 Aug, 2002)

"Whitehats Network Security Resource" URL: http://www.whitehats.com (10 Aug, 2002)

"Microsoft TechNet" URL: http://www.microsoft.com/technet (15 Aug, 2002)

So, Hee. "GCIA Practical Assignment v3.0". (Feb 16, 2002)

Drew, Steven. "GCIA Practical Assignment v3.1" . (Feb, 2002)

Holstein, Michael. "GCIA Practical Assignment v3.1" . (Feb, 2002)

Northcutt, Stephen; Cooper, Mark; Fearnow, Matt and Frederick, Karen. "Intrusion Signatures and Analysis – First Edition". New Riders, 2001