Intrusion Detection In Depth

GCIA Practical Assignment

Version 3.1

Shomiron Das Gupta, CISSP

Table Of Contents

SECTION I – State Of Intrusion Detection

Analysis of a Console Framework

1.1 Abstract

This paper introduces a novel concept in the "art" of intrusion detection. Security analysts and administrators have been developing automated systems that can out-perform manually managed networks. The job of supervising multiple networks belonging to different enterprise clients was becoming more and more difficult. It was becoming impossible for a small watch party of professionals to manually perform the routine tasks of studying the logs and the traces that are gathered each day, ticking events here and circling out events of active targeting, or intending a specific objective.

This paper proposes to lay down a design for an ***Intrusion Detection System Suite****, which aims to automate as many processes as possible so that intrusion analysis becomes easier, faster and consequently more efficient, than what it presently is. The concept being enunciated is the outcome of personal frustration of the author and is based upon his needs in this profession. The model has been developed with open source tools and has been integrated into an architecture, which facilitates data collection and analysis. The model however is semi automated and requires administrator intervention to integrate two groups of automated activities.*

Furthermore, the paper also discusses five attacks in detail, and analyzes logs gathered over five days, objectively.

Mission Statement:
Last but not the least, the author hopes that this paper serves as an inspiration to the fraternity of security personnel to aggrandize the quality and quantity of work being carried out in the area of intrusion detection, presently.

1.2 Introduction

Technology in the 21st century has metamorphosed the way people talk, shop expand and construct businesses in their daily lives. The primary cause of this, being the speed at which communication takes place today. The entire world is networked like never before.

Online businesses doing real time transactions have been on the rise thus having fostered an entire novel community of business entrepreneurs.

Simultaneously on the flip side of this progress is the amount of espionage and stealing that is going on. Information is being stolen. Money is being transferred. Networks are being jammed. Looking closely at these operations that are being carried out by slick professionals, it is not ironic that the same businesses face the danger of losing profits and growth prospects due to loss of customer confidence. This is where network security has gained prominence. With time, it has developed as a science and today is practiced as an art! Besides techniques and tools, a security personnel's hunch has a place in detecting malicious network probes.

The author has spent sleepless nights observing and analyzing suspicious network traffic. Monitoring and following leads often becomes a daunting task when the size of the network being monitored grows out of proportions. The result has been the rising need for a system that would automate some routine tasks in intrusion detection.

The security personnel have to develop strategies to keep his network free of inherent system vulnerabilities and to ward off virus and worm attacks. Such an action requires an understanding of preventive and curative measures.

At the same time, he must also prevent breakdowns and down times caused due to network intrusions. Detecting and stopping network intrusions is a long process in itself. One would need to study traffic patterns, host patterns and should know the difference between suspicious traffic and legitimate traffic. The analyst will need to have the skill to circle out malicious traffic from the wild.

Intrusion Detection Systems are becoming the corner stone of Internet security. These have been deployed the world over to provide the much needed malicious activity reporting to the administrators. Administrators faced the harrowing tasks of capturing packets from the network and trying to analyze traffic patterns and, only in some cases, being successful in picking up network detects.

Today, it is reasonably easy to detect a network probe without having to gape at voluminous packet logs. ID (Intrusion Detection) systems have evolved over the years to bring relief to the security administrator and security analyzers. There a several types of ID systems, which perform the task of analyzing packet logs and detecting the hosts probing the network thus providing relief to the administrators and security analyzers.

There are two major categories of ID systems – Host based Intrusion Detection Systems (HIDS) and Network based Intrusion Detection Systems (NIDS). The HIDS resides on the host and monitors the system comprehensively (network, processes, applications, etc.) to locate any anomaly in the normal operations. On the other hand, NIDS sits on the network and sniffs the network for packages that might contain malicious content.

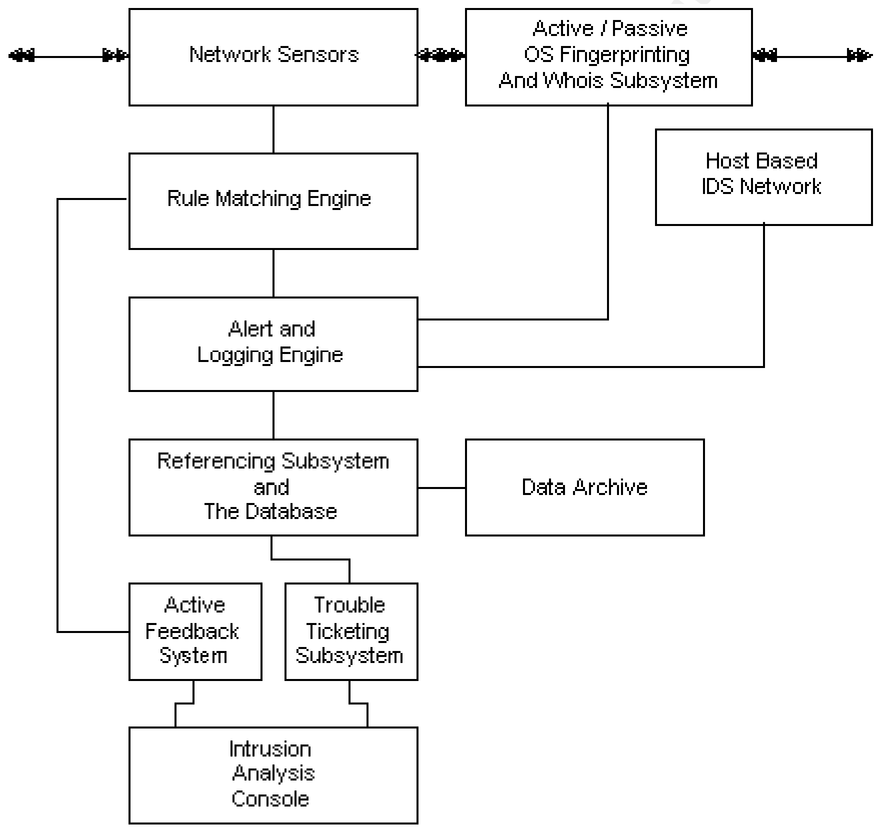
For more information visit the SANS IDS FAQ at
http://www.sans.org/newlook/resources/IDFAQ/ID_FAQ.htm

The first section of this paper, proposes to discuss the architecture with a block-by-block explanation of the activities that each block is designed to perform. This is followed with a working model of the discussed architecture with sample reports and tips on implementation. The subsequent discussions on the process flow for an ID analyst would bring out the merit of this architecture in organizing ID analysis. In the second section of the paper, five detects have been studied. The author has tried to be very analytical in his approach of the subject. And finally the third section is an analysis of five days of log.

This paper is intended to be read by a casual reader as well as experts who wish to automate many processes in Intrusion detection. Industry experts may view the initial architecture and jump straight to section 1.4, which displays and discusses sample reports to demonstrate what can be expected from the system.

1.3 The Framework

Given below is the block representation of the framework that is proposed and discussed in this paper. The double-headed arrows passing through the upper blocks represent the network that is being scanned by the system.


Network Sensors
Active / Passive OS Fingerprinting And Whois Subsystem
Host Based IDS Network
Rule Matching Engine
Alert and Logging Engine
Referencing Subsystem and The Database
Data Archive
Active Feedback System
Trouble Ticketing Subsystem
Intrusion Analysis Console

Along the course of this discussion, the importance of each block and its precise function will be discussed with reference to the block.

1.3.1 Packet Grabber / Sensors

The first block is the ***Packet Grabber*** or more colloquially called the *Network Sensor*. This is a device, which has its network interface set in the promiscuous mode. This mode enables the interface to capture both, packets that are addressed to the interface as well as packets that are not addressed to the interface. This enables the sensor to have a full view of the network. Packets are indiscriminately captured and sent to the rule-matching engine, which is the next logical block. The packet grabber can be visualized as someone who is sitting in a telephone exchange and listening to or eavesdropping all the voice channels handled by the exchange.

The points to be considered while choosing or effectively installing a Network Sensor are:

- **INTERFACE SPEED**: The Interface must capture all the traffic passing through the network. As such, the speed of the interface is an important consideration. If the speed with which the interface is capturing packets is lesser then the total amount of traffic flowing through the network, then the NIDS will not be effective to fullest extent
- **SENSOR PLACEMENT**: The sensor must be placed strategically in the network to enable it to get a full view of the network. If this is not so, the sensor might miss out some traffic passing through it, which also makes the NIDS ineffective.

To learn more on the subject of sensor placement visit
http://www.securityhorizon.com/whitepapers/technical/IDSplace.html

1.3.2 Rule Matching Engine

There are two logical functions that a rule-matching engine needs to perform:

- Receive the captured packets and decode the packet
- Run a pattern matching or a string-matching algorithm to detect anomalies in the packet.

The network sensors send complete packets with the payload to the engine, which in turn disintegrates the packet into its component parts. E.g. a TCP packet is broken down into the frame header, the IP datagram header, the TCP header and finally the payload itself.

A rule base of any ID system is a number of specific patterns that cannot be allowed. The vendor typically develops these rules after studying events that can prove to be disastrous to a network or a host. Events can be anything like system vulnerabilities, implementation vulnerabilities, erroneous TCP, UDP, ICMP, IP packets etc. Generally a rule checks only a particular part of the packet e.g. the TCP header, IP header or the payload. This is done because ID systems need customization to function optimally. Given below is an example of a NIDS rule from Snort.

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"DOS Jolt attack"; fragbits: M;
dsize:408; reference:cve,CAN-1999-0345; classtype:attempted-dos; sid:268; rev:2;)
```

After the packets are decoded from the network, they are matched with the rule base. If there is a match an alert is passed on to the next logical block, which is the alert and logging engine. The process goes as follows:

1.) A rule is picked form the rule base, lets assume it's a rule that checks for errors in the TCP header
2.) The TCP header of the stripped packet is matched with the rule
3.) If there is no match, the process is aborted and the next rule is picked from the rule base
4.) If there is a match, then an alert is generated and sent to the alert and logging engine

Given below are samples of a rule with a packet and an alert generated using Snort. These samples may not be linked.

Sample of a Rule

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"DOS Jolt attack"; fragbits: M;
dsize:408; reference:cve,CAN-1999-0345; classtype:attempted-dos; sid:268; rev:2;)
```

Sample of a Packet

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

[**] spp_frag2: Oversized fragment, probably DoS [**]
06/28-03:43:53.472989 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x2B
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
08 00 00 00 00 00 00 00 00                        .........

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

Sample of an Alert

```
[**] [113:1:1] spp_frag2: Oversized fragment, probable DoS [**]
06/28-03:43:53.496015 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x2B
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
```

1.3.3 Active / Passive OS Fingerprinting and Whois Subsystem

OS Fingerprinting is a technique that is used to determine the OS of the remote host. This technique is widely used for reconnaissance purposes. Most operating systems on the Internet use the TCP/IP protocol. Even though they use the same protocol there are subtle differences in their TCP/IP implementation. This means that the TCP/IP protocol stack response or behavior is OS dependent in most cases, although this need not be a certainty. For example, the default **Time To Live** (TTL) field is set to 32 in a packet coming from a machine running Windows NT server, while the TTL field is set to 64 on a packet coming from a machine running Linux 2.2.x. These minor changes are well cataloged in the OS fingerprinting software, and operating systems are remotely identified on the basis of this data.

Active OS fingerprinting is a technique where a host probes the remote host, evaluates its response and identifies its OS. This technique is called active OS fingerprinting because the questioning host initiates the probe.

An active OS fingerprinting system can further probe the host to obtain additional information like open ports on the system, uptime of the system, time zone settings etc. Following is an example of active OS fingerprinting.

```
# nmap -sS -O xxx.yyy.zzz.aaa

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on  (xxx.yyy.zzz.aaa):
(The 1532 ports scanned but not shown below are in state: closed)
Port       State       Service
7/tcp      open        echo
9/tcp      open        discard
13/tcp     open        daytime
17/tcp     open        qotd
19/tcp     open        chargen
21/tcp     open        ftp
25/tcp     open        smtp
26/tcp     open        unknown
42/tcp     open        nameserver
53/tcp     open        domain
80/tcp     open        http
110/tcp    open        pop-3
119/tcp    open        nntp
135/tcp    open        loc-srv
139/tcp    open        netbios-ssn
443/tcp    open        https
445/tcp    open        microsoft-ds
563/tcp    open        snews
1025/tcp   open        listen
1026/tcp   open        nterm
1433/tcp   open        ms-sql-s
6666/tcp   open        irc-serv

Remote OS guesses: Windows Me or Windows 2000 RC1 through final release, MS
Windows2000 Professional RC1/W2K Advance Server Beta3, Windows Millenium Edition
v4.90.3000

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

For more information on active OS fingerprinting visit http://www.insecure.org

Passive OS fingerprinting is a flip of active OS fingerprinting. In such a system, the fingerprinting software picks up packet streams from the network and tries to identify the source operating systems. Passive OS fingerprinting systems work very much like a network sensor since it picks ups data streams from the network, analyses the source packets and tries to draw parallels to the pre-built operating system stack signatures.

In this architecture this block proves to be a handy reconnaissance system. The block can produce a lot of valuable information like:

1. Operating system of the remote host
2. Ports open on the remote host
3. System uptime and even time

4. Zone settings in some cases

Most importantly, this block can immediately find out if the remote system is alive or not. If the remote system is not responding, then it clearly proves that the remote IP address has been spoofed. Such a network is prey to active targeting. This is one of the most obvious signs of reconnaissance before a system compromise attempt.

The passive OS fingerprinting software identifies the operating systems of all the hosts that have an established connection with the network, whereas, the active OS fingerprinting software probes only those hosts who have appeared in alerts from the ID system. The output of both the passive and the active OS fingerprinting system is passed on to the alerts and logging subsystem.

The **Whois Subsystem** works one step further. It picks up an IP address that has generated an alert from the alerting engine and probes the whois servers for data on the IP. The whois servers can provide information like the net-block size and owner of the block and the contact addresses of the block. This can be handy information when solving geographically defined problems. The information from the whois server is then passed on to the alerts and logging engine for further processing.

1.3.4 Alerts and Logging Engine

The alert and logging engine is a central point for accumulating feedback from the active systems such as the Rule-Matching Engine, The Active / Passive OS Fingerprinting System and the Whois Subsystem. This unit may have the plugins for producing alerts in different formats and to insert the alerts into the database and the referencing subsystem. The logging engine simply captures the packets with anomalies so that an analyst can later analyze them closely. Several plugins are available for interfacing between standard ID output formats and various database systems. Thus, it is very easy to enable databases on most of the ID systems around prevalent.

```
[**] [1:1287:3] WEB-IIS scripts access [**]
[Classification:  sid] [Priority: 2]
06/28-03:11:54.238279 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x39F
My.Net.42.6:2078  ->  209.73.225.14:80  TCP  TTL:128  TOS:0x0  ID:25829  IpLen:20
DgmLen:913 DF
***AP*** Seq: 0xACF44358  Ack: 0xAD8E6A6  Win: 0x4470  TcpLen: 20
```

Above is an example of an alert generated by Snort a NIDS.

1.3.5 Host Based IDS Network

Network ID systems have been in extensive use in the past few years as it is a centralized approach to ID. It is rather unfortunate to see network administrators neglect the potential in HIDS systems. However, the combination of NIDS and HIDS is potentially the best. Although HIDS systems are expensive to deploy over a large network, they must be deployed over networks running critical services.

HID Systems come in different flavors; there are some HID systems that just detect anomalous behavior and there are others that stop suspicious activity. These components are more part of the defense armory rather then the detection armory. Sophisticated HID systems can be deployed and be monitored over the network. They can be configured to report to a central console. This enables the analyst to view events in a single console.

Situations where the HID systems go beyond the budgetary allocations, one could opt for some open source tools to perform the role of system monitoring. These tools can then be clubbed together to report to a central unit.

In this architecture, we are trying to club the HID systems network to report to the alerts and logging engine where alerts generated by the HID systems can be documented by the central alert and logging engine and then can be passed on in a formatted manner to the database and the referencing subsystem.

1.3.6 Referencing Subsystem and the Database

Referencing is the trickiest portion of the activity of putting together reports from several sources. The referencing subsystem picks up alerts and logs from the alerts and logging engine. Units like the Rule Matching Engine, The Active / Passive Os Fingerprinting System and the HIDS network actually generate these reports. Subsequently, the referencing subsystem matches the reports and finds referencing points between the different reports.

Several referencing points could be considered to bring all the different reports together. Time could be used as a referencing point, a primary key could be assigned to each event which in turn could be used as a referencing point or finally the IP addresses itself could be used as referencing points.

Referencing with time is a little tricky because packets travel at a very high speed in a network. Even if there is a difference of a fraction of seconds, the time referencing system will fail as there would be a difference between the timestamp on packets coming from different sensors or say different systems. For example, there might be a difference in the timestamps of a packet captured by the sensor and the recon info that comes from the Active / Passive OS Fingerprinting System. This will create an error in referencing the two packets even though they share the same event. It is difficult to use a primary key as the referencing point since it is difficult to assign the same key to events that are generated by different systems.

It is therefore, safer to us the IP address in question as a referencing point. The IP address can be used as a reference for the network sensor that captured it, the Active / Passive OS Fingerprinting System, the Whois Subsystem and also the HID systems. The use of IP address based referencing system is illustrated in the following sections:

The database system is the heart of the entire analysis system. The database stores all the records that generated alerts, the recon results, the alerts that are generated from the HID systems etc. Thus, the database is a crucial component since it would be required in every process of correlation.

The selected database for such systems must possess the properties given under:

- It must be fast, since there will be very quick inputs coming from different sources It must be able to handle large volume of records
- It must be robust as a failure could be fatal
- It should be without vulnerabilities as wanting sources could attack it

1.3.7 Trouble Ticketing

The trouble-ticketing engine generates the analyst's front end. It picks up alerted data from the database and produces the trouble tickets for the analyst. Trouble tickets are information slips that describe an alert with all the needed information to initiate a detailed analysis. Trouble tickets are very useful as they can give historical information on the particular incident. Tickets are useful in trend analysis to identify hosts with suspicious behavior. Trouble tickets can be provided on different reference points. The IP address again makes the best referencing coordinate. Trouble tickets can also be sorted according the various parameters. Alerts can be sorted according to time period, alert generated, source address, destination address etc.

For example a trouble ticket sorted by IP should provide information about the incident, statistical information, the other incidents in which the said IP address is involved and the priority of the attacks. There are several systems that are available and can be implemented on the standard ID systems. These systems can be web based or can have an application front end.

Note that formats of trouble tickets can vary from analyst to analyst, for this depends on the route of analysis for an analyst.

1.3.8 Automated Response

Following the generation of a trouble ticket is the generation of a response coupon that marks a particular host (host of interest) and monitors all its activities at high priority. Response coupons can also introduce a new rule or an access list to deny any access to the host to prevent damage.

For example when an analyst notices suspicious connections to his network, he can immediately trigger a response by marking the host such that every time the host initiates a connection to this network, all the activities automatically get red lined. In an extreme case, the response system can generate a rule for the firewall that will prevent any connections into the network initiated by the host.

1.3.9 Data Archiving

Data archiving is a very simple archiving unit that archives all records older than a particular number of days. Large networks generate large amount of logs storage of these log files often become a very resource intensive problem. Erasing the log files is also not an option because of historical evidence or even host or vulnerability evaluation may be required at a later date.

Ideally log files can be maintained in their original state for analysis till there is cessation of activity from the particular host. However, it is prudent to preserve the log files forever. Log files can be archived after they have been reviewed. Archiving slows the retrieval process. It is difficult to retrieve an entry for a particular host after it has been archived.

1.3.10 Intrusion Analysis Console

The analysis console is the simplest of blocks in the architecture. This is generally a central computer that has access to all the logs and alerts generated by the deployed ID systems. This block should preferably have live feeds from the trouble ticketing system and should also have some mechanism of alerting the analyst of high priority activity. Some of the innovative alerting mechanisms are audible alerts, email alerts and paging systems.

In short an analysis console should give a birds-eye view of the ID system that is being monitored, and provide details as and when required by the analyst to investigate any issue.

1.4 A Working Architecture

A simple working model of the discussed architecture is demonstrated in this section. A rudimentary structure has been designed using available open source tools and some bit of scripting. The sections are split up according to tool functions rather block functions in the architecture.

A review of the reports that can be generated to aid intrusion analysis will also be made. This, however, is not the most efficient way of implementing the architecture. More effective integration of tools is possible with more skillful programming.

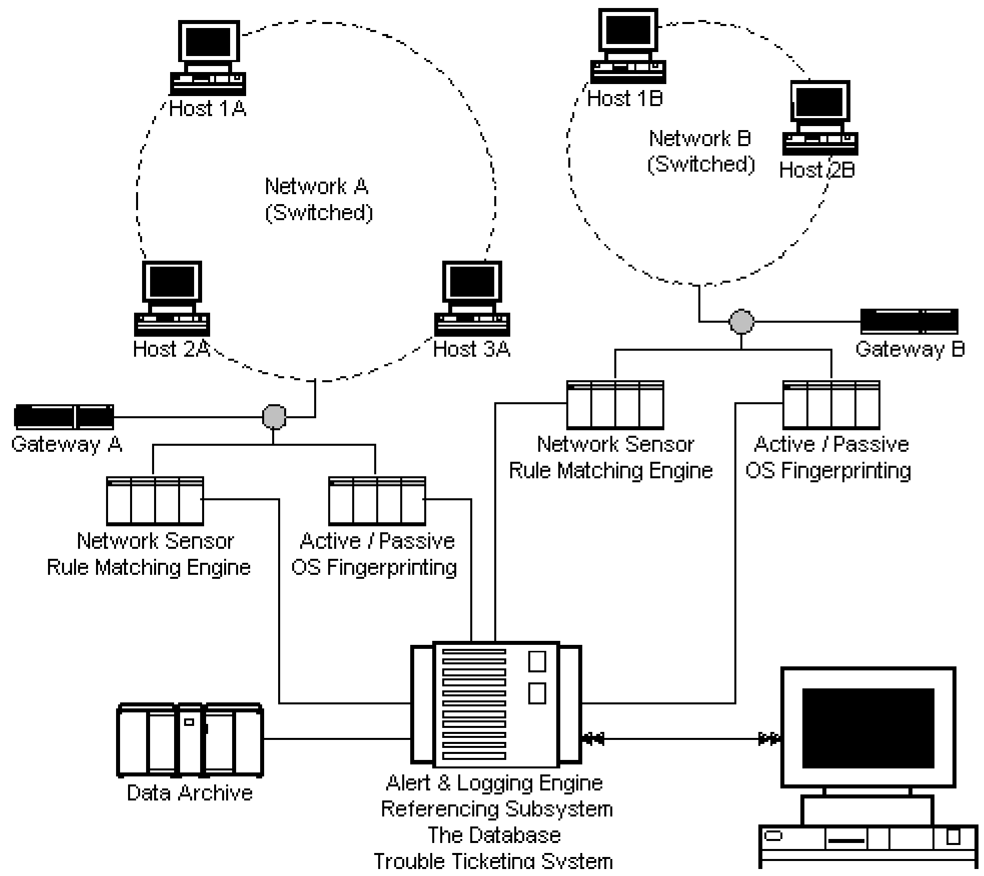
1.4.1 Tools for the Operation

All the tools used in the design are open source and are available free.

- The basic network ID uses SNORT 1.8.6. This is easy to deploy and there are several available output plugins for it to choose from. Its rule set is editable and makes the software customizable and versatile.
- NMAP and P0F are used for Active And Passive OS Fingerprinting respectively
- Both these tools are customizable and provide options that allow the user great flexibility
- MYSQL is used as the database MYSQL is robust and is freely available on the net.
- Simple tools and scripting languages that are fairly easy to understand and follow have been used in odd pockets to create the chain. There are some tools that may be unheard of but this is not a cause to worry. Everything has been adequately explained in the following sections.

1.4.2 The System

Given below is a diagrammatic representation of the architecture that has been discussed in the sections above. The diagram shows two networks viz. Network A and Network B being monitored by NID Systems. Note this system doesn't have a representation for the HID system network. The HID system can also be integrated with the system below using scripted interface or a standard data exchange format.



1.4.3 The Network Sensors / Rule Matching Engine

Snort is one of the most useful tools in intrusion detection and analysis. It has a useful feature set like network sensor, packet de-assembler / re-assembler, rule matching engine, logging engine, alerting engine, plugins for several kinds of logging systems etc. Moreover, Snort is an open source tool and is available for free.

Snort is deployed in the two networks above to capture the packets, perform the rule matching process and generate alerts. As discussed network sensors have to be deployed in such a way that it has a view of all the traffic coming in and going out of the network. Since the network is a switched network, the port to the gateway is mirrored to the network senor. The mirroring concept has been explained in the following section.

Hubs and Switches

Deploying a sensor on a network using hubs is very easy because hubs are non-intelligent devices and it will simply replicate the sender data to all the ports. Capturing packets is easy because the sensor will be sent packets even when they are not targeted towards it. In a switched network, each port will specifically receive packets that are destined for the host connected to the port. In this case the sensor doesn't get the full view of the traffic passing through the network. To make a sensor effective on a switched network, one must ideally replicate the main entrance and exit points to the network. This can be achieved by enabling a feature called port mirroring on the switch. For more information on getting Snort to work on switched networks visit http://www.snort.org/docs/faq.html

The network sensors pick up each packet from the network, the de-assemblers rips the packet to separate all the headers and the payloads. The rule-matching engine then analyzes the ripped part of the packet. In case there is a match, an alert is generated. This alert and the packet that created the alert are then sent to the logging facility.

Both the sensors must have Snort installed with the latest rule set and can be initiated using the following command:

```
# snort -de -l /soft/logs/ -c /soft/rules/snort.conf -D -s
```

Explanation of the command line options

```
Usage: snort [-options] <filter options>
-d           Dump the Application Layer
-e           Display the second layer header info
-D           Run Snort in background (daemon) mode
-s           Log alert messages to syslog
-l <ld>      Log to directory <ld>
-c <rules>   Use Rules File <rules>
```

In this scenario, Snort has to be configured to log into a central database. This is achieved by configuring the output plugins. Snort can be configured using a variety of output plugins.

Given below is an example of configuring the Snort output plugins to log all the generated alerts into a MYSQL database. This is done in the snort.conf file that is been used as the rules file, generally can be found in the rules directory.

```
output database: alert, mysql, user=snortuser password=snortpass dbname=snort
host=dbsrv
```

1.4.4 Active / Passive OS Fingerprinting

A tool called p0f does passive OS fingerprinting. p0f is an open source tool available for free. It sits on a promiscuous device and monitors the packets that travel through the network. All the attributes of the packet are captured and matched with its fingerprinting database. The following command initiates p0f and writes into a file called 'pos.txt'

```
# p0f > pos.txt
```

After every defined unit time say five minutes, this txt file can be purged and all the information gathered is updated into a table in the database. A sample output of p0f is given below:

```
# p0f
p0f: passive os fingerprinting utility, version 1.8.2
(C) Michal Zalewski <lcamtuf@gis.net>, William Stearns <wstearns@pobox.com>
p0f: file: '/etc/p0f.fp', 150 fprints, iface: 'eth0', rule: 'all'.
203.94.229.30 [7 hops]: Linux 2.2.9 - 2.2.18
203.94.229.30 [7 hops]: Linux 2.2.9 - 2.2.18
202.68.145.90 [10 hops]: Linux 2.4.2 - 2.4.14 (1)
203.94.229.30 [7 hops]: Linux 2.2.9 - 2.2.18
203.94.229.30 [7 hops]: Linux 2.2.9 - 2.2.18
203.94.229.30 [7 hops]: Linux 2.2.9 - 2.2.18
203.94.229.30 [7 hops]: Linux 2.2.9 - 2.2.18
203.94.229.30 [7 hops]: Linux 2.2.9 - 2.2.18
203.94.229.30 [7 hops]: Linux 2.2.9 - 2.2.18
202.149.213.49 [10 hops]: CacheOS 3.1 on a CacheFlow 6000
My.Net.42.6 [1 hops]: Windows 2000 (9)
203.94.229.30 [7 hops]: Linux 2.2.9 - 2.2.18
202.149.212.218 [10 hops]: CacheOS 3.1 on a CacheFlow 6000
202.149.212.218 [10 hops]: CacheOS 3.1 on a CacheFlow 6000
202.149.212.218 [10 hops]: CacheOS 3.1 on a CacheFlow 6000
202.149.212.218 [10 hops]: CacheOS 3.1 on a CacheFlow 6000
203.94.229.30 [7 hops]: Linux 2.2.9 - 2.2.18
203.200.8.149: UNKNOWN [1024:57:1460:0:0:1:1:64].
203.94.229.30 [7 hops]: Linux 2.2.9 - 2.2.18
```

The Active OS fingerprinting unit picks up all the addresses that have generated alerts from the database and runs a probe on them host. The objective of the process of probing is obtain information of the host system, such as open ports, host OS, and determines if the host is reachable. The host not being reachable is a clear sign of the source address being spoofed. All these activities can be easily performed using NMAP. The process goes as follows:

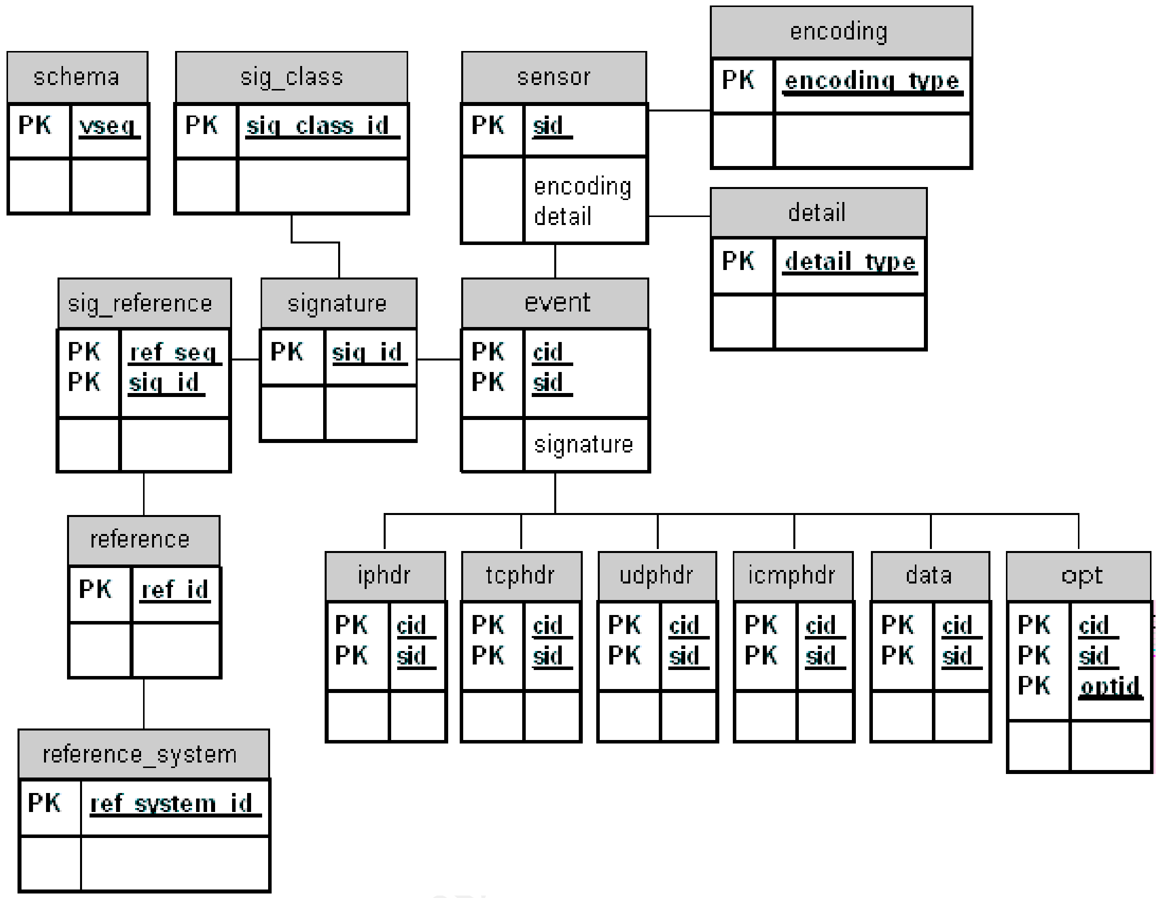
- A list of IP addresses are created by running a query on the database every unit time say two minutes or so
- To find information required for analysis the analyst must list the information he requires from the system and must run Nmap accordingly.

This can be achieved by writing a script that picks targets from the file and saves the output in the database. At this moment the objective is to find out if the host can be reached. As such, the current script simply inserts 'Host is UP' in the database if the host is reachable and it inserts 'Host is Down' when the host is not reachable. Nmap is used in this example to find out if the source was spoofed.

Note the time interval should be as short as possible because sometimes hosts are momentarily forced to being unreachable by running a denial of service attack. After the source is spoofed in the attack the DOS is lifted and the unreachable system comes back into action. This status may not be detected if the time interval is long and an active fingerprinting is done after the host becomes reachable.

1.4.5 Alert, Logging Engine and The Database

In this setup, both the sensors log into a central MYSQL database server in which records can be inserted using remote hosts. In this case the Snort plugins are configured to insert data from the sensors into the database. The database schema is given below with explanation of the important tables.



In the above schema, the tables 'event' and the table 'signature' are of utmost importance. The 'event' table provides the primary key for the alert generation. The 'signature' table gives the name of the alert or the description about the alert. The 'iphdr', 'tcphdr', 'udphdr' and the 'icmphdr' tables contain the header fields of the corresponding alerts from the 'event' table. The above schema is a remake of the schema available at http://www.snort.org/docs/snortdb.png

Following is a customized table that has a consolidated view on every IP address that has generated an alert. This table is called 'con' and it can be created using the script given below:

```
CREATE TABLE con ( timestamp                    DATETIME NOT NULL,
                   cid                INT                UNSIGNED NOT NULL,
                   sid                INT                UNSIGNED NOT NULL,
                   sip                INT                UNSIGNED NOT NULL,
                   dip                INT                UNSIGNED NOT NULL,
                   sig                VARCHAR(255)  NOT NULL,
                   priority     INT          UNSIGNED,
                   pos                VARCHAR(100)  NOT NULL,
                   aos                VARCHAR(100)  NOT NULL);
```

The field description for the table 'con' are given below:
timestamp – Contains the date and time of the alert
cid – Contains a referencing key for data in other snort tables
sid – Contains a referencing key for data in other snort tables
sip – Contains the source IP address
dip – Contains the destination IP address
sig – Contains the description of the alert
priority – Contains the priority of the alert
pos – Passive OS fingerprinting information
aos – Active OS fingerprinting information

This table will pull in data from the snort tables and other input sources like the Active / Passive OS Finger Printing System and create a consolidated information table.

Note: Integration of the HID systems is not yet over. As such, concrete output samples are not as yet provided. Web based whois systems are used to pull information on a particular IP address. This is done if the analyst requests it from the trouble ticket display.

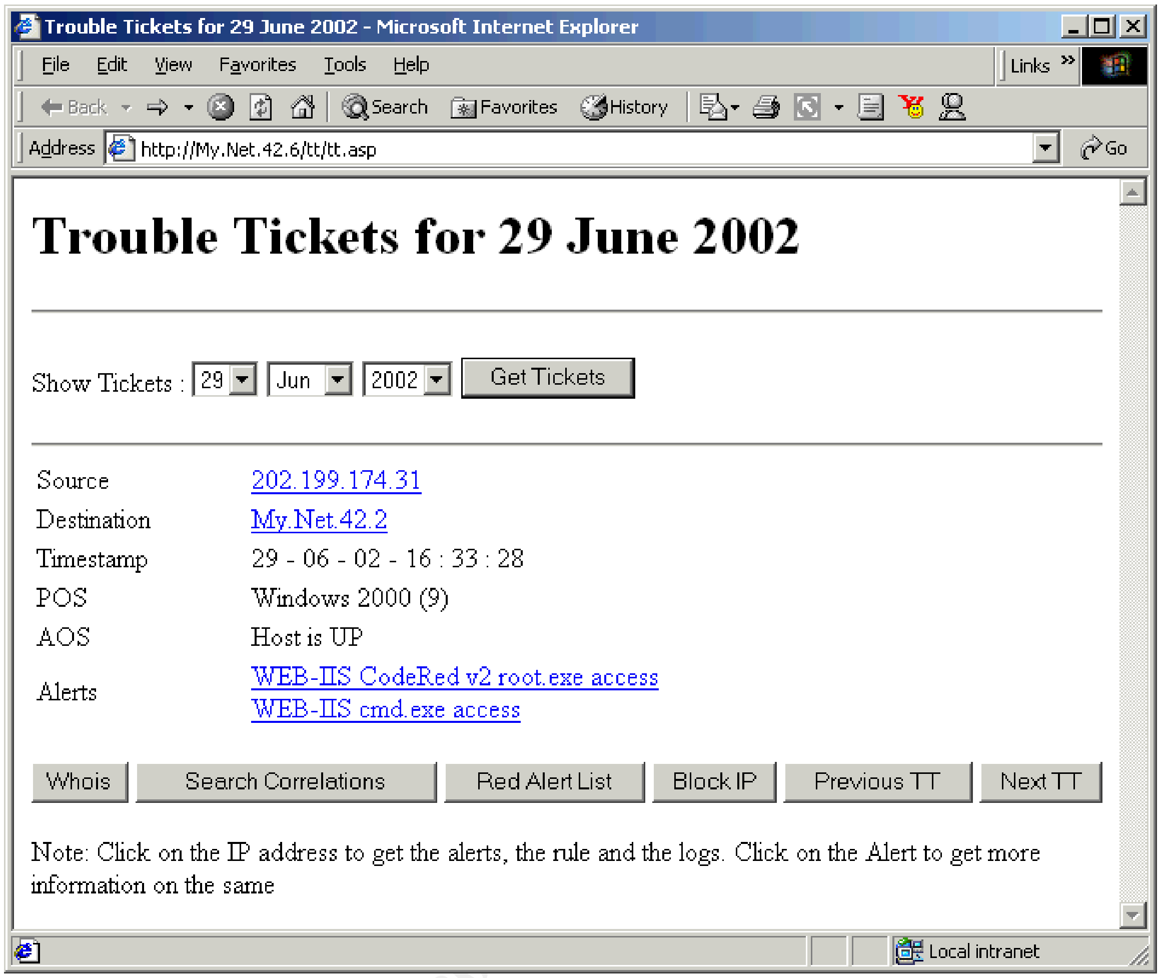
Following is a query and its output from the con table. This query only pulls out events with priority = 1.

```
mysql> select distinct inet_ntoa(sip), inet_ntoa(dip), sig, pos, aos from con
where priority = 1;
+-----------------+-----------------+-------------------------------------+------------------
+--------------+
| inet_ntoa(sip)  | inet_ntoa(dip)  | sig                                 | pos
| aos          |
+-----------------+-----------------+-------------------------------------+------------------
+--------------+
| 202.183.185.74  | My.Net.42.15    | WEB-IIS cmd.exe access              |
Windows 2000 (9) | Host is UP   |
| 202.199.174.31  | My.Net.42.2     | WEB-IIS CodeRed v2 root.exe access |
Windows 2000 (9) | Host is UP   |
| 202.199.174.31  | My.Net.42.2     | | Windows 2000 (9) | Host is UP   |
| 202.110.198.106 | My.Net.42.14    | WEB-IIS CodeRed v2 root.exe access |
Windows 2000 (9) | Host is UP   |
| 193.252.42.13   | My.Net.42.13    | WEB-IIS CodeRed v2 root.exe access |
Windows 2000 (9) | Host is UP   |
| 193.252.42.13   | My.Net.42.13    | WEB-IIS cmd.exe access              |
Windows 2000 (9) | Host is UP   |
| 202.194.21.65   | My.Net.42.2     | WEB-IIS CodeRed v2 root.exe access |
Windows 2000 (9) | Host is UP   |
| 202.194.21.65   | My.Net.42.2     | WEB-IIS cmd.exe access              |
Windows 2000 (9) | Host is UP   |
| 202.44.42.246   | My.Net.42.2     | WEB-IIS CodeRed v2 root.exe access |
Windows 2000 (9) | Host is UP   |
| 202.44.42.246   | My.Net.42.2     | WEB-IIS cmd.exe access              |
Windows 2000 (9) | Host is UP   |
| My.Net.42.16    | 216.136.145.152 | WEB-ATTACKS mail command attempt    |
Windows 2000 (9) | Host is UP   |
| 202.101.35.170  | My.Net.42.11    | WEB-IIS CodeRed v2 root.exe access |
Windows 2000 (9) | Host is DOWN |
| 202.101.35.170  | My.Net.42.11    | WEB-IIS cmd.exe access              |
Windows 2000 (9) | Host is DOWN |
| 202.101.35.170  | My.Net.42.14    | WEB-IIS CodeRed v2 root.exe access |
Windows 2000 (9) | Host is DOWN |
| 202.101.35.170  | My.Net.42.14    | WEB-IIS cmd.exe access             |
Windows 2000 (9) | Host is DOWN |
| 202.194.23.84   | My.Net.42.13    | WEB-IIS CodeRed v2 root.exe access |
Windows 2000 (9) | Host is UP   |
| 202.194.23.84   | My.Net.42.13    | WEB-IIS cmd.exe access             |
Windows 2000 (9) | Host is UP   |
+-----------------+-----------------+------------------------------------+------------------
+--------------+
```

1.4.6 Trouble Tickets and Active Response

The trouble ticketing system pulls out events from the table 'con' from the database. This is easy because the database holds all the data required for generating a trouble ticket. A single consolidated ticket is generated for all the alerts on a particular IP address. This consolidated window helps the analyst to understand the different attack techniques in case of active targeting. The ticket should have links to all the parts of the packet that generated the alert, i.e. the entire logged packet. This helps the analyst to understand the exact format of the packet. Understanding packets is very important as it provides exact explanation for crafted packets or newer attacks.

Following is a screen shot of a sample trouble ticketing system. This is a web-based system that pulls out trouble tickets from the database. It precisely displays only required information for an apt overview and response system. It displays the source and destination IP addresses. Clicking on the IP addresses will give you more information on the IP address and the protocol being used. The interface also provides an option to view the packet itself. This initial screen provides description of all the alerts that have been generated by the IP address, the timestamp of the alert, the Passive and Active OS Fingerprinting output and a link to the whois sites that provide more information on the IP and the net block. There is also a facility to search previous correlations on the IP address or the kind of attack.


Trouble Tickets for 29 June 2002 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
Back Search Favorites History
Address http://My.Net.42.6/tt/tt.asp
Go
Show Tickets : 29 Jun 2002 Get Tickets
Whois
Search Correlations
Red Alert List
Block IP
Previous TT
Next TT
Local intranet

Trouble Tickets for 29 June 2002

Source	202.199.174.31
Destination	My.Net.42.2
Timestamp	29 - 06 - 02 - 16 : 33 : 28
POS	Windows 2000 (9)
AOS	Host is UP
Alerts	WEB-IIS CodeRed v2 root.exe access WEB-IIS cmd.exe access

Note: Click on the IP address to get the alerts, the rule and the logs. Click on the Alert to get more information on the same

The active response system is a part of the trouble ticket itself. There is an option to provide a red alert signal every time there is activity from the particular IP address. This just adds the IP address to a table in the database containing the IP address and the corresponding alerts. It also adds a rule in the snort rule base that comes up with an alert the moment there is activity from this IP address.

The second option though is to directly block the IP address. This system can write a rule for an IP chain packet filter that prevents the IP address from entering the network or the host. In case the network uses any other firewall or filtering technique a simple list of blocked IP address can be generated and be sent to the firewall administrator to update the access control lists.

SECTION II– Network Detects

2.1 Detect # 1 - Covert Channel Detection

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

06/28-20:29:39.095630 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x5C
203.94.237.146 -> My.Net.42.6 ICMP TTL:57 TOS:0x0 ID:0 IpLen:20 DgmLen:78 DF
Type:0  Code:0  ID:27165  Seq:0  ECHO REPLY
75 73 65 20 74 68 65 20 6C 69 6E 6B 73 20 61 6E  use the links an
64 20 74 68 65 20 70 61 73 73 77 6F 72 64 73 20  d the passwords
61 6E 64 20 63 20 77 68 61 74 20 75 20 77 61 6E  and c what u wan
74 0A                                            t.

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

06/28-20:29:40.293877 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x3C
203.94.237.146 -> My.Net.42.6 ICMP TTL:57 TOS:0x0 ID:0 IpLen:20 DgmLen:29 DF
Type:0  Code:0  ID:27165  Seq:0  ECHO REPLY
0A                                               .

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

06/28-20:29:56.914638 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x44
203.94.237.146 -> My.Net.42.6 ICMP TTL:57 TOS:0x0 ID:0 IpLen:20 DgmLen:54 DF
Type:0  Code:0  ID:27165  Seq:0  ECHO REPLY
68 74 74 70 3A 2F 2F 32 30 32 2E 38 37 2E 34 32  http://My.Net.42
2E 32 38 2F 73 74 75 66 66 0A                    .28/stuff.

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

06/28-20:30:06.825548 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x3C
203.94.237.146 -> My.Net.42.6 ICMP TTL:57 TOS:0x0 ID:0 IpLen:20 DgmLen:45 DF
Type:0  Code:0  ID:27165  Seq:0  ECHO REPLY
75 73 65 72 6E 61 6D 65 3A 20 68 6F 6E 6E 65 72  username: honner
0A                                               .

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

06/28-20:30:19.634118 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x3F
203.94.237.146 -> My.Net.42.6 ICMP TTL:57 TOS:0x0 ID:0 IpLen:20 DgmLen:49 DF
Type:0  Code:0  ID:27165  Seq:0  ECHO REPLY
70 61 73 73 77 6F 72 64 3A 20 68 61 78 6F 77 6F  password: haxowo
72 6B 32 32 0A                                   rk22.

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

06/28-20:30:20.097772 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x3C
203.94.237.146 -> My.Net.42.6 ICMP TTL:57 TOS:0x0 ID:0 IpLen:20 DgmLen:29 DF
Type:0  Code:0  ID:27165  Seq:0  ECHO REPLY
0A                                               .

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/28-20:31:09.035930 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x6A
203.94.237.146 -> My.Net.42.6 ICMP TTL:57 TOS:0x0 ID:0 IpLen:20 DgmLen:92 DF
Type:0  Code:0  ID:27165  Seq:0  ECHO REPLY
68 74 74 70 3A 2F 2F 32 30 32 2E 38 37 2E 34 32  http://My.Net.42
2E 32 39 2F 73 74 75 66 66 20 28 62 61 63 6B 75  .29/stuff (backu
70 20 73 72 76 20 75 73 75 61 6C 6C 79 20 68 61  p srv usually ha
73 20 61 6C 6C 20 74 68 65 20 70 69 63 73 29 0A  s all the pics).

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

06/28-20:31:13.789261 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x3C
203.94.237.146 -> My.Net.42.6 ICMP TTL:57 TOS:0x0 ID:0 IpLen:20 DgmLen:45 DF
Type:0  Code:0  ID:27165  Seq:0  ECHO REPLY
75 73 65 72 6E 61 6D 65 3A 20 68 6F 6E 6E 65 72  username: honner
0A                                               .

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

06/28-20:31:27.734856 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x40
203.94.237.146 -> My.Net.42.6 ICMP TTL:57 TOS:0x0 ID:0 IpLen:20 DgmLen:50 DF
Type:0  Code:0  ID:27165  Seq:0  ECHO REPLY
70 61 73 73 77 6F 72 64 3A 20 68 61 78 6F 6D 61  password: haxoma
6E 69 61 32 32 0A                                nia22.

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

06/28-20:31:28.617112 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x3C
203.94.237.146 -> My.Net.42.6 ICMP TTL:57 TOS:0x0 ID:0 IpLen:20 DgmLen:29 DF
Type:0  Code:0  ID:27165  Seq:0  ECHO REPLY
0A                                               .

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

06/28-20:31:41.745656 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x47
203.94.237.146 -> My.Net.42.6 ICMP TTL:57 TOS:0x0 ID:0 IpLen:20 DgmLen:57 DF
Type:0  Code:0  ID:27165  Seq:0  ECHO REPLY
69 20 73 75 67 67 65 73 74 20 67 65 74 20 74 68  i suggest get th
65 20 70 69 63 73 20 66 69 72 73 74 0A           e pics first.

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

06/28-20:31:42.484552 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x3C
203.94.237.146 -> My.Net.42.6 ICMP TTL:57 TOS:0x0 ID:0 IpLen:20 DgmLen:29 DF
Type:0  Code:0  ID:27165  Seq:0  ECHO REPLY
0A                                               .

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

06/28-20:31:47.590394 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x3C
203.94.237.146 -> My.Net.42.6 ICMP TTL:57 TOS:0x0 ID:0 IpLen:20 DgmLen:43 DF
Type:0  Code:0  ID:27165  Seq:0  ECHO REPLY
63 61 74 63 68 20 79 61 20 6C 61 74 65 72 0A     catch ya later.

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

06/28-20:31:49.832166 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x3C
203.94.237.146 -> My.Net.42.6 ICMP TTL:57 TOS:0x0 ID:0 IpLen:20 DgmLen:32 DF
```

```
Type:0  Code:0  ID:27165  Seq:0  ECHO REPLY
62 79 65 0A                                  bye.

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

06/28-20:31:53.133774 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x72
203.94.237.146:32772 -> My.Net.42.15:22 TCP TTL:57 TOS:0x10 ID:56945 IpLen:20 DgmLen:100 DF
***AP*** Seq: 0xBFC11F5E  Ack: 0xAA991087  Win: 0x2280  TcpLen: 32
TCP Options (3) => NOP NOP TS: 152458 7243697
DF 1D 9A DE 01 81 55 73 93 FE 88 70 F7 82 E1 31  ......Us...p...1
3B 92 C5 70 73 DE 71 6D CA AA 18 1C DA C7 BF F1  ;..ps.qm........
0E EA 96 DF C9 8A EE A6 F8 0E 02 6B C7 56 41 01  ...........k.VA.

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

2.1.1 Source of trace

The trace is from my own network. I was monitoring ICMP traffic, moving through my network. For this purpose, I had left tcpdump running all night by mistake. I noticed several packets from a single host with varying Datagram lengths (DgmLen). This is rather unusual in a case when the host is a PC and not a network device. This hit the eye because when there were no ICMP Echo Requests made to the host why was it sending ICMP Echo Replies.

2.1.2 Detect was generated by

Detect was captured using tcpdump. Traffic was captured using the command below:

```
# tcpdump -w detect.log -s 65535
```

Explanation:

-w	Writes captured data into a file
-s	Snapshot length. Where default is 68

Complete help on tcpdump is available (`# tcpdump --help`)

Later, the captured log was analyzed using snort.

```
# snort -dev -r detect.log icmp and host = My.Net.42.15
```

Explanation:

-d	Dumps application layer
-e	Displays second layer header info
-v	Be verbose
-r	Read and process tcpdump file
icmp and host My.Net.42.15	This is a filter that picks up icmp packets from host My.Net.42.15

Complete help on snort is available (`# snort -h`)

2.1.3 Probability of the source address being spoofed

Probability of the source address being spoofed is low because the system was used for two-way covert communication. Moreover, an active OS fingerprinting script-using nmap automatically ran on the host and discovered that the host was up.

Note similar attack or communication could happen even with a spoofed IP address, provided there is a prior understanding between the two hosts.

2.1.4 Description of attack

This attack was an implementation of an ICMP Covert Communication Channel between two hosts. This kind of covert channels is employed to communicate between hosts through a firewall. Most administrators choose to allow ICMP traffic into the network. Covert channels can be used like a chat session to communicate between two hosts.

More advanced software can use an ICMP covert channel that can be bound to a shell and can be used to run shell commands on a compromised host. This is one communication that easily escapes detection.

The attack came from a dialup user connected to MTNL (a service provider in India). Given under is the whois lookup of the IP address.

```
inetnum:      203.94.224.0 - 203.94.255.255
netname:      MTNL
descr:        Mahanagar Telephone Nigam Ltd., ISP Division, New Delhi
descr:        Planning Development and Operation of Telecom. Services.
country:      IN
admin-c:      DC59-AP
tech-c:       DC59-AP
mnt-by:       APNIC-HM
changed:      hostmaster@apnic.net 19981224
source:       APNIC
```

2.1.5 Attack Mechanism

The attack uses an ICMP Echo Reply to transfer information from one host to the other. The data travels in the payload of the ICMP Echo Reply packet. The sender sends the ICMP Echo Replies to the receiving host with the data in the ICMP payload. The receiver is initiated to pick up all the ICMP packets from the sender host. Once the receiver picks up ICMP packets from the sender, it decodes the information in the payload and displays it on the screen.

Tools for covert channel communication are available. Tools performing multiple functions using covert channel communication are also available. Covert channel communication is commonly implemented or performed using ICMP, and using UDP with some advanced tools.

2.1.6 Correlations

http://rr.sans.org/covertchannels/covert_shells.php
Finest information material on Covert Channels and tools is available at rr.sans.org Written by J. Christian Smith.

http://online.securityfocus.com/archive/101/254298
There is a very short discussion available at Security Focus Online.

2.1.7 Evidence of Active Targeting

Active targeting is obvious because these covert channels have to be initiated by the attacker so that it can communicate. These channels cannot be initiated automatically by a scanning script or by a worm.

2.1.8 Severity

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

Criticality = 2 (Non targeted attack)
Lethality = 3 (Social engineering has lots of destructive potential)
System Countermeasures = 2 (ICMP is enabled on the target)
Network Countermeasures = 2 (ICMP is allowed to the target)

Severity = (2 + 3) – (2 + 2)
Severity = 1

2.1.9 Defensive recommendation

ICMP is a very important protocol and most administrators would like to implement it. Whenever, ICMP poses a serious threat, one could filter ICMP traffic at the firewall itself and prevent ICMP Echo Reply packets from surpassing the firewall.

2.1.10 Multiple Choice Test Question

Q. What is the telltale sings of an ICMP Covert Channel?
1. Multiple SYN packets are received to initiate the connection
2. Size of the Datagram keeps varying
3. All Echo Reply packets carry covert channel information
4. A packet with its URG bit set

Answer: 2

2.2 Detect # 2 – IP Fragmentation Attack

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

06/29-11:55:04.735020 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x3C
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
08 00 00 00 00 00 00 00 00                       .........

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

06/29-11:55:04.735040 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x3C
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
08 00 00 00 00 00 00 00 00                       .........

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

06/29-11:55:04.735044 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x3C
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
08 00 00 00 00 00 00 00 00                       .........

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

06/29-11:55:04.735064 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x3C
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
08 00 00 00 00 00 00 00 00                       .........

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

06/29-11:55:04.735068 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x3C
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
08 00 00 00 00 00 00 00 00                       .........

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

06/29-11:55:04.735088 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x3C
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
08 00 00 00 00 00 00 00 00                       .........

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

06/29-11:55:04.735092 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x3C
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
08 00 00 00 00 00 00 00 00                       .........

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

06/29-11:55:04.735113 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x3C
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
08 00 00 00 00 00 00 00 00                       .........
```

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

06/29-11:55:04.735117 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x3C
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
08 00 00 00 00 00 00 00 00                       .........

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

06/29-11:55:04.735137 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x3C
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
08 00 00 00 00 00 00 00 00                       .........

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

06/29-11:55:04.735141 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x3C
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
08 00 00 00 00 00 00 00 00                       .........

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

06/29-11:55:04.735161 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x3C
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
08 00 00 00 00 00 00 00 00                       .........

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

2.2.1 Source of trace

This trace was picked up from my network. This attack performed a DOS on a newly installed Windows NT server. After looking into the Snort alerts I found the following trace:

```
[**] [113:1:1] spp_frag2: Oversized fragment, probable DoS [**]
06/28-03:43:53.462741 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x2B
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
```

The trace indicated that there was a fragmentation attack that Snort couldn't detect. Tcpdump with snap length set to 65535 was used to capture traffic into a file.

2.2.2 Detect was generated by

Originally the attack was detected by Snort version 1.8.6 (Build 105). But further investigation required tcpdump to capture traffic and again snort to analyze the captured traffic.

The following commands were used to capture traffic and then to analyze the traffic:

```
# tcpdump –w detect.log –s 65535
```

Explanation:

-w	Writes captured data into a file
-s	Snapshot length. Where default is 68

Complete help on tcpdump is available (`# tcpdump --help`)

The captured log was later analyzed using snort

```
# snort -dev -r detect.log host = 127.0.0.1
```

Explanation:

-d	Dumps application layer
-e	Displays second layer header info
-v	Be verbose
-r	Read and process tcpdump file
host = 127.0.0.1	This is a filter that picks up packets from host 127.0.0.1

Complete help on snort is available (`# snort -h`)

2.2.3 Probability the source address was spoofed

The source was definitely spoofed as the packets had 127.0.0.1 (loopback) address in the source field. The actual source IP could not be identified because of the spoof.

2.2.4 Description of attack

The attack is a DOS attack. This attack pushes the device into an unstable state, which can stop device response or device crashing. This is one of the known flaws in IP and most vendors have devised ways to address the problem. Following are the alerts that are generated using Snort Version 1.8.6 (Build 105).

```
[**] [113:1:1] spp_frag2: Oversized fragment, probable DoS [**]
06/28-03:43:53.462741 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x2B
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B

[**] [1:528:3] BAD TRAFFIC loopback traffic [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
06/28-03:43:53.462741 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x2B
127.0.0.1 -> My.Net.42.234 ICMP TTL:255 TOS:0x0 ID:1109 IpLen:20 DgmLen:29
Frag Offset: 0x1FFE   Frag Size: 0xFFFFE00B
```

Snort does not have an exact rule for this attack but it does recognize it to be an IP fragmentation attack. Moreover this attack spoofs the source IP to be 127.0.0.1 (loopback), this generates one more alert that says this is bad and unusual traffic.

2.2.5 Attack Mechanism

The attack sends crafted packets to the destination host at a very high speed. These packets exploit the IP fragmentation concept. Whenever packets are too large for the transport medium, they are divided into smaller parts at the starting point of the route. The packets are temporarily written to the memory. At the end of the route, they are re-assembled into original packet.

This attack sends fragmented packets to the target host. The packets are similar and have their "more fragments" bit set indicating thereby that there are more packets in the chain. The receiving device, (in this case the Windows NT server) writes each packet to its memory till it receives the last packet of the chain. The machine eventually runs out of memory in storing the incoming fragmented packets. This causes the machine to go into an unstable state, which results in a DOS.

There are several tools available to conduct this attack. Experienced attackers can also use packet generators with scripts to generate crafted packets in a very fast-multithreaded loop. This attack mode can escape ID systems that have unique signatures, but are still effective.

2.2.6 Correlations

The Jolt and Jolt2 are attacks that are known to do similar kind of damage. I haven't tried them out myself but read some information about them. Following are some links that define similar attacks.

Great description of jolt2
http://www.securiteam.com/exploits/Jolt2_-_a_new_Windows_DoS_attack.html

Good thread that discusses Jolt2 again.
http://lists.insecure.org/win2ksecadvice/2000/May/0056.html

Vulnerability note found on CERT site.
http://www.kb.cert.org/vuls/id/35958

Bugtraq of the IP fragmentation attack on Checkpoint Firewall – 1
http://online.securityfocus.com/bid/1312/info/

2.2.7 Evidence of active targeting

Active targeting is clear in this case as this attack has to be initiated by an attacker and cannot be accomplished by a worm or a virus. No correlations can be found as the source IP was spoofed. Therefore, one cannot say if the attacker conducted scans prior to this attack or not.

2.2.8 Severity

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

Criticality = 4 (The server installed was to be a Business server)
Lethality = 4 (DOS attack)
System Countermeasures = 1 (No OS updates were installed)
Network Countermeasures = 0 (No firewall was in place)

Severity = (4 + 4) – (1 + 0)
Severity = 7

2.2.9 Defensive recommendation

Microsoft has released a patch for almost all its operating systems
Following are the links for the same

Windows 95:
http://download.microsoft.com/download/ win95/update/8070/w95/EN-US/259728USA5.EXE

Windows 98:
http://download.microsoft.com/download/ win98/update/8070/w98/EN-US/259728USA8.EXE

Windows NT 4.0 Workstation, Server and Server, Enterprise Edition:
http://www.microsoft.com/Downloads/Release.asp?ReleaseID=20829

Windows NT 4.0 Server, Terminal Server Edition:
http://www.microsoft.com/Downloads/Release.asp?ReleaseID=20830

Windows 2000 Professional, Server and Advanced Server:
http://www.microsoft.com/Downloads/Release.asp?ReleaseID=20827

Checkpoint's Firewall-1 is also known to be vulnerable to the attack. The link that explains the vulnerability and provides a solution to reduce the risk is given below:.
http://www.checkpoint.com/techsupport/alerts/ipfrag_dos.html

2.2.10 Multiple choice test question

Q. In an IP Fragmentation attack?
1. The DF (Don't Fragment) Bit is set
2. The More Fragment Bit is set
3. Only TCP is used
4. ICMP is never used

Answer: 2

2.3 Detect # 3 – DDOS MSTREAM Client to Handler

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

03/26-02:49:17.368823 0:A0:24:A8:7B:BD -> 0:10:A4:EC:54:B8 type:0x800 len:0x3C
192.168.0.56:60038 -> 192.168.0.200:15104 TCP TTL:40 TOS:0x0 ID:64924 IpLen:20
DgmLen:40
******S* Seq: 0xD3D72E8F  Ack: 0x0  Win: 0x400  TcpLen: 20

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

03/26-02:49:17.370076 0:10:A4:EC:54:B8 -> 0:A0:24:A8:7B:BD type:0x800 len:0x3C
192.168.0.200:15104 -> 192.168.0.56:60038 TCP TTL:128 TOS:0x0 ID:17057 IpLen:20
DgmLen:40
***A*R** Seq: 0x0  Ack: 0xD3D72E90  Win: 0x0  TcpLen: 20

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

[**] DDOS mstream client to handler [**]
03/26-02:49:17.368823 0:A0:24:A8:7B:BD -> 0:10:A4:EC:54:B8 type:0x800 len:0x3C
192.168.0.56:60038 -> 192.168.0.200:15104 TCP TTL:40 TOS:0x0 ID:64924 IpLen:20
DgmLen:40
******S* Seq: 0xD3D72E8F  Ack: 0x0  Win: 0x400  TcpLen: 20

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

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

03/26-02:53:23.267738 0:A0:24:A8:7B:BD -> 0:10:A4:DF:4B:2A type:0x800 len:0x3C
192.168.0.56:57725 -> 192.168.0.1:15014 TCP TTL:52 TOS:0x0 ID:46835 IpLen:20
DgmLen:40
******S* Seq: 0x3223EEB7  Ack: 0x0  Win: 0x400  TcpLen: 20

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

03/26-02:53:23.274116 0:10:A4:DF:4B:2A -> 0:A0:24:A8:7B:BD type:0x800 len:0x3C
192.168.0.1:15014 -> 192.168.0.56:57725 TCP TTL:128 TOS:0x0 ID:37064 IpLen:20
DgmLen:40
***A*R** Seq: 0x0  Ack: 0x3223EEB8  Win: 0x0  TcpLen: 20

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

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

03/26-05:23:36.721879 0:E0:98:2:54:DD -> 0:60:97:85:42:2F type:0x800 len:0x3C
192.168.0.53:24965 -> 192.168.0.5:15104 TCP TTL:64 TOS:0x0 ID:19546 IpLen:20
DgmLen:44
******S* Seq: 0x11965293  Ack: 0x0  Win: 0x200  TcpLen: 24
TCP Options (1) => MSS: 1460

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

03/26-05:23:36.721951 0:60:97:85:42:2F -> 0:E0:98:2:54:DD type:0x800 len:0x3C
192.168.0.5:15104 -> 192.168.0.53:24965 TCP TTL:255 TOS:0x0 ID:24960 IpLen:20
DgmLen:40
***A*R** Seq: 0x0  Ack: 0x11965294  Win: 0x0  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] DDOS mstream client to handler [**]
03/26-05:23:36.721879 0:E0:98:2:54:DD -> 0:60:97:85:42:2F type:0x800 len:0x3C
192.168.0.53:24965 -> 192.168.0.5:15104 TCP TTL:64 TOS:0x0 ID:19546 IpLen:20
DgmLen:44
******S* Seq: 0x11965293  Ack: 0x0  Win: 0x200  TcpLen: 24
TCP Options (1) => MSS: 1460

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

2.3.1 Source of trace

These detects were picked up from the SANS 2000 hack fest logs found on the Snort site http://www.snort.org/packets.html. The signature looks very innocuous but is one of the most dangerous attacks in the business.

2.3.2 Detect was generated by

These detects were generated using Snort Version 1.8.6 (Build 105). The command below was used to generate alerts from the logs.

```
# snort -de -r detect.log -l /var/log/snort -c /snort/rules/snort.conf
```

Explanation:

-d	Dumps application layer
-e	Displays second layer header info
-r	Read and process tcpdump file
-l	Generates alerts and packet logs and saves them in the log directory
-c	Path to the snort rules file snort.conf

Complete help on snort is available (`# snort -h`)

2.3.3 Probability the source address was spoofed

The probability of this attack being spoofed is extremely low because the attacker is trying to initiate a TCP connection TCP connections are difficult to spoof because they need the three-way handshake to function.

2.3.4 Description of attack

This attack is a Distributed Denial Of Service (DDOS) attack. DDOS is by far the most lethal attack possible. DDOS attacks work in multiple layers-some in two and others in three layers. A DDOS attack consists of hundreds of hosts attacking a single target host or network. A DOS (Denial Of Service) is caused either by exhausting the system resources of the target host or by exhausting the network resources of the target network.

In a three-layer DDOS attack there is a master that controls some hundred odd hosts called handlers. Each handler in turn controls another hundred odd hosts called agents.

Agents are the ones that actually attack the target host or the network. This means a massive attack consisting of a million or so hosts can be triggered and controlled by just one master host.

Detects given below have picked from hack fest logs, host 192.168.0.56 and host 192.168.0.53 probe for the handler, Trojan on the target host to respond. The exact process of attack is described in the attack mechanism section.

```
[**] [1:249:1] DDOS mstream client to handler [**]
[Classification: Attempted Denial of Service] [Priority: 2]
03/26-02:49:17.368823 0:A0:24:A8:7B:BD -> 0:10:A4:EC:54:B8 type:0x800 len:0x3C
192.168.0.56:60038 -> 192.168.0.200:15104 TCP TTL:40 TOS:0x0 ID:64924 IpLen:20
DgmLen:40
******S* Seq: 0xD3D72E8F  Ack: 0x0  Win: 0x400  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS111]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0138]

[**] [1:249:1] DDOS mstream client to handler [**]
[Classification: Attempted Denial of Service] [Priority: 2]
03/26-02:53:14.456853 0:A0:24:A8:7B:BD -> 0:10:A4:DF:4B:2A type:0x800 len:0x3C
192.168.0.56:57725 -> 192.168.0.1:15104 TCP TTL:52 TOS:0x0 ID:32859 IpLen:20
DgmLen:40
******S* Seq: 0x3223EEB7  Ack: 0x0  Win: 0x400  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS111]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0138]

[**] [1:249:1] DDOS mstream client to handler [**]
[Classification: Attempted Denial of Service] [Priority: 2]
03/26-05:23:36.721879 0:E0:98:2:54:DD -> 0:60:97:85:42:2F type:0x800 len:0x3C
192.168.0.53:24965 -> 192.168.0.5:15104 TCP TTL:64 TOS:0x0 ID:19546 IpLen:20
DgmLen:44
******S* Seq: 0x11965293  Ack: 0x0  Win: 0x200  TcpLen: 24
TCP Options (1) => MSS: 1460
[Xref => http://www.whitehats.com/info/IDS111]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0138]
```

Above were the mstream alerts generated by Snort after analyzing the logs from hack fest.

2.3.5 Attack mechanism

The mstream attack is a complex attack. It has a three-layered cascading attack structure - the intruder, the handler and the agent. The intruder controls the handlers and the handlers in turn control the agents. The agents are the ones who attack.

The intruder makes a connection to the handler on TCP port 6723 and the handler connects the agent on UDP port 7983. In this system the agent sends feedback to the handler on UDP port 9325.

It is important to note the following operational:

- The handler doesn't require root privileges but the agent requires root privileges to carry out attacks
- The intruder can easily alter port numbers at compile time

- In most cases, the handler and the agent running as rpc.wall in binary form
- Agents generally conduct their attacks by sending crafted packets
- Agent binaries contain the list of handlers that are defined by the intruder at compile time

All information provided above is a summary of an Incident Note from CERT. For more information and better reading http://www.cert.org/incident_notes/IN-2000-05.html

When the intruder triggers the attack, the agents launch a TCP ACK flood against the target host or network. Following is a sample of the crafted TCP ACK Flood packets coming form the agent. The sample has also been picked up from the CERT incident note.

```
11:58:43.531109 eth0 > xxx.xxx.xxx.xxx.2458 > 10.1.1.2.51479: .
2110392958:2110392958(0) ack 0 win 16384 [tos 0x8] (ttl 255, id 12979)
11:58:43.531116 eth0 > xxx.xxx.xxx.xxx.2714 > 10.1.1.3.29405: .
2127170174:2127170174(0) ack 0 win 16384 [tos 0x8] (ttl 255, id 13235)
11:58:43.531136 eth0 > xxx.xxx.xxx.xxx.2970 > 10.1.1.2.29837: .
2143947390:2143947390(0) ack 0 win 16384 [tos 0x8] (ttl 255, id 13491)
```

In our case host's 192.168.0.56 has made an attempt to connect to host 192.168.0.200 and 192.168.0.1 and host 192.168.0.53 has made an attempt to connect to host 192.168.0.5. The destination ports have been changed to 15104 to avoid detection. This is the connection between the intruder and the handler because that is the only connection that uses TCP to connect.

2.3.6 Correlations

David Dittrich, George Weaver, Sven Dietrich and Neil Long have done the most detailed analysis of mstream, The paper discusses every aspect of mstream and is available at http://staff.washington.edu/dittrich/misc/mstream.analysis.txt

Incident note IN-2000-05 from CERT describes mstream the Distribute Denial Of Service attack tool. http://www.cert.org/incident_notes/IN-2000-05.html

A Bugtraq message actually gives the code for master.c and server.c this message is available at http://marc.theaimsgroup.com/?l=bugtraq&m=95715370208598&w=2

Alert generated by Internet Security Systems gives operative details on mstream and is available at http://bvlive01.iss.net/issEn/delivery/xforce/alertdetail.jsp?id=advise48

CAN-2000-0138 from the ICAT Metabase listed in the CVE vulnerability database. The alert also has the links to the CVE note itself http://icat.nist.gov/icat.cfm?cvename=CAN-2000-0138

2.3.7 Evidence of active targeting

Active targeting is clear because the host's 192.168.0.56 and 192.168.0.53 were conducting a vulnerability scan on few of the other hosts in the network. This is not possible by a worm activity. Hence, it will be classified as a case of active targeting.

2.3.8 Severity

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

Criticality = 2 (Not sure about the host, would have been a user desktop at least)
Lethality = 4 (DDOS attack)
System Countermeasures = 0 (Not sure if the trojan was present in the system)
Network Countermeasures = 2 (No network countermeasure was in place)

Severity = (2 + 4) – (0 + 2)
Severity = 4

2.3.9 Defensive recommendation

The two aspects to the problem are:-

- The intruder installing an agent or a handler
- The intruder making a connection to the handler or a handler making a connection to the agent

The solution to the first problem is to keep the machines up with all the patches and service packs to prevent an intruder from gaining shell access to the target host. A shell will enable him to install the binaries for the handler or the agent. A firewall or a filtering router should be employed to allow requests for selected services

A firewall or a filtering router should be used to drop connections for known Trojan ports. This will disallow intruders from probing for handler access to the network. Robust security practices have to be followed to minimize the risk or denial of service attacks. Some good information and samples on improving security practices can be found at http://www.cert.org/security-improvement/. One also has to proactively upgrade one's knowledge on recent denial of service techniques and practices.

2.3.10 Multiple choice test question

Q. Difference between a DOS and a DDOS?
1. DOS is a many to many attack unlike DDOS
2. There is no difference its just jargon
3. DDOS is a uses TCP, DOS uses ICMP
4. DOS is a single layer attack architecture unlike DDOS

Answer: 4

2.4 Detect # 4 – Netbios NT NULL Session

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

03/25-06:51:34.281782 0:10:A4:EC:54:B8 -> 0:10:A4:EC:54:E0 type:0x800 len:0xEA
192.168.0.200:1267 -> 192.168.0.210:139 TCP TTL:128 TOS:0x0 ID:52888 IpLen:20
DgmLen:220 DF
***AP*** Seq: 0x108981  Ack: 0xEB38  Win: 0x21CF  TcpLen: 20
00 00 00 B0 FF 53 4D 42 73 00 00 00 00 18 03 80  .....SMBs.......
00 00 CE C9 13 66 3E 64 76 94 00 00 00 00 FE CA  .....f>dv.......
00 00 00 00 0D 75 00 84 00 04 11 32 00 00 00 00  .....u.....2....
00 00 00 01 00 00 00 00 00 00 00 D4 00 00 00 47  ...............G
00 00 00 00 00 00 57 00 69 00 6E 00 64 00 6F 00  ......W.i.n.d.o.
77 00 73 00 20 00 4E 00 54 00 20 00 31 00 33 00  w.s. .N.T. .1.3.
38 00 31 00 00 00 00 00 57 00 69 00 6E 00 64 00  8.1.....W.i.n.d.
6F 00 77 00 73 00 20 00 4E 00 54 00 20 00 34 00  o.w.s. .N.T. .4.
2E 00 30 00 00 00 00 00 04 FF 00 00 00 00 00 01  ..0.............
00 21 00 00 5C 00 5C 00 54 00 41 00 52 00 47 00  .!..\.\.T.A.R.G.
45 00 54 00 5C 00 49 00 50 00 43 00 24 00 00 00  E.T.\.I.P.C.$...
49 50 43 00                                      IPC.

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

[**] NETBIOS NT NULL session [**]
03/25-06:51:34.281782 0:10:A4:EC:54:B8 -> 0:10:A4:EC:54:E0 type:0x800 len:0xEA
192.168.0.200:1267 -> 192.168.0.210:139 TCP TTL:128 TOS:0x0 ID:52888 IpLen:20
DgmLen:220 DF
***AP*** Seq: 0x108981  Ack: 0xEB38  Win: 0x21CF  TcpLen: 20
00 00 00 B0 FF 53 4D 42 73 00 00 00 00 18 03 80  .....SMBs.......
00 00 CE C9 13 66 3E 64 76 94 00 00 00 00 FE CA  .....f>dv.......
00 00 00 00 0D 75 00 84 00 04 11 32 00 00 00 00  .....u.....2....
00 00 00 01 00 00 00 00 00 00 00 D4 00 00 00 47  ...............G
00 00 00 00 00 00 57 00 69 00 6E 00 64 00 6F 00  ......W.i.n.d.o.
77 00 73 00 20 00 4E 00 54 00 20 00 31 00 33 00  w.s. .N.T. .1.3.
38 00 31 00 00 00 00 00 57 00 69 00 6E 00 64 00  8.1.....W.i.n.d.
6F 00 77 00 73 00 20 00 4E 00 54 00 20 00 34 00  o.w.s. .N.T. .4.
2E 00 30 00 00 00 00 00 04 FF 00 00 00 00 00 01  ..0.............
00 21 00 00 5C 00 5C 00 54 00 41 00 52 00 47 00  .!..\.\.T.A.R.G.
45 00 54 00 5C 00 49 00 50 00 43 00 24 00 00 00  E.T.\.I.P.C.$...
49 50 43 00                                      IPC.

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

[**] NETBIOS NT NULL session [**]
03/25-06:51:34.281782 0:10:A4:EC:54:B8 -> 0:10:A4:EC:54:E0 type:0x800 len:0xEA
192.168.0.200:1267 -> 192.168.0.210:139 TCP TTL:128 TOS:0x0 ID:52888 IpLen:20
DgmLen:220 DF
***AP*** Seq: 0x108981  Ack: 0xEB38  Win: 0x21CF  TcpLen: 20
00 00 00 B0 FF 53 4D 42 73 00 00 00 00 18 03 80  .....SMBs.......
00 00 CE C9 13 66 3E 64 76 94 00 00 00 00 FE CA  .....f>dv.......
00 00 00 00 0D 75 00 84 00 04 11 32 00 00 00 00  .....u.....2....
00 00 00 01 00 00 00 00 00 00 00 D4 00 00 00 47  ...............G
00 00 00 00 00 00 57 00 69 00 6E 00 64 00 6F 00  ......W.i.n.d.o.
77 00 73 00 20 00 4E 00 54 00 20 00 31 00 33 00  w.s. .N.T. .1.3.
38 00 31 00 00 00 00 00 57 00 69 00 6E 00 64 00  8.1.....W.i.n.d.
6F 00 77 00 73 00 20 00 4E 00 54 00 20 00 34 00  o.w.s. .N.T. .4.
2E 00 30 00 00 00 00 00 04 FF 00 00 00 00 00 01  ..0.............
```

```
00 21 00 00 5C 00 5C 00 54 00 41 00 52 00 47 00  .!..\.\.T.A.R.G.
45 00 54 00 5C 00 49 00 50 00 43 00 24 00 00 00  E.T.\.I.P.C.$...
49 50 43 00                                      IPC.

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

03/25-07:03:36.174989 0:10:A4:EC:54:B8 -> 0:10:A4:EC:54:E0 type:0x800 len:0xEA
192.168.0.200:1308 -> 192.168.0.210:139 TCP TTL:128 TOS:0x0 ID:48799 IpLen:20
DgmLen:220 DF
***AP*** Seq: 0x108F28  Ack: 0xECFB  Win: 0x21CF  TcpLen: 20
00 00 00 B0 FF 53 4D 42 73 00 00 00 00 18 03 80  .....SMBs.......
00 00 CE C9 13 66 3E 64 76 94 00 00 00 00 FE CA  .....f>dv.......
00 00 00 00 0D 75 00 84 00 04 11 32 00 00 00 00  .....u.....2....
00 00 00 01 00 00 00 00 00 00 00 D4 00 00 00 47  ...............G
00 00 00 00 00 00 57 00 69 00 6E 00 64 00 6F 00  ......W.i.n.d.o.
77 00 73 00 20 00 4E 00 54 00 20 00 31 00 33 00  w.s. .N.T. .1.3.
38 00 31 00 00 00 00 00 57 00 69 00 6E 00 64 00  8.1.....W.i.n.d.
6F 00 77 00 73 00 20 00 4E 00 54 00 20 00 34 00  o.w.s. .N.T. .4.
2E 00 30 00 00 00 00 00 04 FF 00 00 00 00 00 01  ..0.............
00 21 00 00 5C 00 5C 00 54 00 41 00 52 00 47 00  .!..\.\.T.A.R.G.
45 00 54 00 5C 00 49 00 50 00 43 00 24 00 00 00  E.T.\.I.P.C.$...
49 50 43 00                                      IPC.

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

[**] NETBIOS NT NULL session [**]
03/25-07:03:36.174989 0:10:A4:EC:54:B8 -> 0:10:A4:EC:54:E0 type:0x800 len:0xEA
192.168.0.200:1308 -> 192.168.0.210:139 TCP TTL:128 TOS:0x0 ID:48799 IpLen:20
DgmLen:220 DF
***AP*** Seq: 0x108F28  Ack: 0xECFB  Win: 0x21CF  TcpLen: 20
00 00 00 B0 FF 53 4D 42 73 00 00 00 00 18 03 80  .....SMBs.......
00 00 CE C9 13 66 3E 64 76 94 00 00 00 00 FE CA  .....f>dv.......
00 00 00 00 0D 75 00 84 00 04 11 32 00 00 00 00  .....u.....2....
00 00 00 01 00 00 00 00 00 00 00 D4 00 00 00 47  ...............G
00 00 00 00 00 00 57 00 69 00 6E 00 64 00 6F 00  ......W.i.n.d.o.
77 00 73 00 20 00 4E 00 54 00 20 00 31 00 33 00  w.s. .N.T. .1.3.
38 00 31 00 00 00 00 00 57 00 69 00 6E 00 64 00  8.1.....W.i.n.d.
6F 00 77 00 73 00 20 00 4E 00 54 00 20 00 34 00  o.w.s. .N.T. .4.
2E 00 30 00 00 00 00 00 04 FF 00 00 00 00 00 01  ..0.............
00 21 00 00 5C 00 5C 00 54 00 41 00 52 00 47 00  .!..\.\.T.A.R.G.
45 00 54 00 5C 00 49 00 50 00 43 00 24 00 00 00  E.T.\.I.P.C.$...
49 50 43 00                                      IPC.

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

03/25-07:15:36.168215 0:10:A4:EC:54:B8 -> 0:10:A4:EC:54:E0 type:0x800 len:0xEA
192.168.0.200:1327 -> 192.168.0.210:139 TCP TTL:128 TOS:0x0 ID:21924 IpLen:20
DgmLen:220 DF
***AP*** Seq: 0x109113  Ack: 0xED7F  Win: 0x21CF  TcpLen: 20
00 00 00 B0 FF 53 4D 42 73 00 00 00 00 18 03 80  .....SMBs.......
00 00 CE C9 13 66 3E 64 76 94 00 00 00 00 FE CA  .....f>dv.......
00 00 00 00 0D 75 00 84 00 04 11 32 00 00 00 00  .....u.....2....
00 00 00 01 00 00 00 00 00 00 00 D4 00 00 00 47  ...............G
00 00 00 00 00 00 57 00 69 00 6E 00 64 00 6F 00  ......W.i.n.d.o.
77 00 73 00 20 00 4E 00 54 00 20 00 31 00 33 00  w.s. .N.T. .1.3.
38 00 31 00 00 00 00 00 57 00 69 00 6E 00 64 00  8.1.....W.i.n.d.
6F 00 77 00 73 00 20 00 4E 00 54 00 20 00 34 00  o.w.s. .N.T. .4.
2E 00 30 00 00 00 00 00 04 FF 00 00 00 00 00 01  ..0.............
00 21 00 00 5C 00 5C 00 54 00 41 00 52 00 47 00  .!..\.\.T.A.R.G.
45 00 54 00 5C 00 49 00 50 00 43 00 24 00 00 00  E.T.\.I.P.C.$...
49 50 43 00                                      IPC.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] NETBIOS NT NULL session [**]
03/25-07:15:36.168215 0:10:A4:EC:54:B8 -> 0:10:A4:EC:54:E0 type:0x800 len:0xEA
192.168.0.200:1327 -> 192.168.0.210:139 TCP TTL:128 TOS:0x0 ID:21924 IpLen:20
DgmLen:220 DF
***AP*** Seq: 0x109113  Ack: 0xED7F  Win: 0x21CF  TcpLen: 20
00 00 00 B0 FF 53 4D 42 73 00 00 00 00 18 03 80  .....SMBs.......
00 00 CE C9 13 66 3E 64 76 94 00 00 00 00 FE CA  .....f>dv.......
00 00 00 00 0D 75 00 84 00 04 11 32 00 00 00 00  .....u.....2....
00 00 00 01 00 00 00 00 00 00 00 D4 00 00 00 47  ...............G
00 00 00 00 00 00 57 00 69 00 6E 00 64 00 6F 00  ......W.i.n.d.o.
77 00 73 00 20 00 4E 00 54 00 20 00 31 00 33 00  w.s. .N.T. .1.3.
38 00 31 00 00 00 00 00 57 00 69 00 6E 00 64 00  8.1.....W.i.n.d.
6F 00 77 00 73 00 20 00 4E 00 54 00 20 00 34 00  o.w.s. .N.T. .4.
2E 00 30 00 00 00 00 00 04 FF 00 00 00 00 00 01  ..0.............
00 21 00 00 5C 00 5C 00 54 00 41 00 52 00 47 00  .!..\.\.T.A.R.G.
45 00 54 00 5C 00 49 00 50 00 43 00 24 00 00 00  E.T.\.I.P.C.$...
49 50 43 00                                      IPC.

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

2.4.1 Source of trace

These detects were picked up from the SANS 2000 hack fest logs found on the Snort site http://www.snort.org/packets.html. Since this has been a very prominent method of performing reconnaissance on Windows NT systems, I thought of including it in my detects.

2.4.2 Detect was generated by

These detects were generated using Snort Version 1.8.6 (Build 105). The command used to generate alerts from the logs was:.

```
# snort -de -r detect.log -l /var/log/snort -c /snort/rules/snort.conf
```

Explanation:

-d	Dumps application layer
-e	Displays second layer header info
-r	Read and process tcpdump file
-l	Generates alerts and packet logs and saves them in the log directory
-c	Path to the snort rules file snort.conf

Complete help on snort is available (`# snort -h`)

The following rule used by Snort, picked up the attack.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS NT NULL session";
flags:A+; content: "|00 00 00 00 57 00 69 00 6E 00 64 00 6F 00 77 00 73 00 20 00
4E 00 54 00 20 00 31 00 33 00 38 00 31|"; reference:bugtraq,1163;
reference:cve,CVE-2000-0347; reference:arachnids,204; classtype:attempted-recon;
sid:530; rev:5;)
```

2.4.3 Probability of the source address being spoofed

The probability of the source address being spoofed is less because the attacker expects a response from the target system and it is unlikely that the attacker would use it like a DOS attack.

2.4.4 Description of attack

This attack affects Microsoft Windows NT Server and Workstation 3.51 and 4.0 Sp3. This is a facility for an anonymous user to extract information from a Windows NT machine. The user can list the domain users and enumerate the shares that the workstation or server has.

Some systems are vulnerable to this attack technique. The attacker exploits this vulnerability and performs a reconnaissance on the system. No special tools are required except the ones shipped with the OS itself.

2.4.5 Attack mechanism

A single domain Windows NT network is able to authenticate accounts connections to list the users and shares of the domain. The server knows exactly who has logged in and where. In this scenario generally everyone has a user account and not an anonymous account.

In the case of a Multiple Domain scenario, domains require anonymous user logins to list domain information such as users and shares. This vulnerability can be easily exploited using the NETUSE command that comes with a Windows NT installation.

There are several programs that exploit this vulnerability to extract information. One example of code that extracts information form a vulnerable system is available at http://online.securityfocus.com/bid/494/exploit/.

2.4.6 Correlations

The attacker was found scanning the whole subnet for hosts responding to the NULL session vulnerability.

Information and exploit code can be found at the bugtraq from securityfocus available at http://online.securityfocus.com/bid/494/exploit/.

Microsoft also has an interesting article that supplies information on the vulnerability. This is available at http://support.microsoft.com/default.aspx?scid=kb;EN-US;q143474.

2.4.7 Evidence of active targeting

There is evidence of active targeting because this seemed to be a part of a scan that was performed on the target host.

2.4.8 Severity

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

Criticality = 2 (Not sure about the host)
Lethality = 3 (Potential information leak)
System Countermeasures = 5 (Assuming the OS was updated)
Network Countermeasures = 0 (No firewall seemed to be in place)

Severity = (2 + 3) – (5 + 0)
Severity = 0

2.4.9 Defensive recommendation

One solution is to disable this feature that is been exploited by going to the following key in the registry: This attack is exploiting a certain Registry feature. It can be prevented by disabling this by going to the following key in the registry:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\LSA

Add Value and use the following entry:
Value Name: RestrictAnonymous
Data Type: REG_DWORD
Value: 1
Restart the computer for changes to take effect. This information was picked up from http://support.microsoft.com/default.aspx?scid=kb;EN-US;q143474.

2.4.10 Multiple choice test question

Q. NULL Sessions
1. Are TCP sessions
2. Are UDP, ICMP sessions
3. Are user sessions without a password
4. Are used in Data Archiving

Answer: 3

2.5 Detect # 5 – Brute Force Attack on SNMP Community String

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

07/06-12:18:14.335436 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x53
203.166.65.22:1076 -> My.Net.42.15:161 UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:69
DF
Len: 49
30 82 00 25 02 01 00 04 06 70 75 62 6C 69 63 A1  0..%.....public.
18 02 01 01 02 01 00 02 01 00 30 0D 30 82 00 09  ..........0.0...
06 05 2B 06 01 02 01 05 00                       ..+......

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

07/06-12:18:14.336066 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x46
My.Net.42.15 -> 203.166.65.22 ICMP TTL:125 TOS:0x0 ID:19922 IpLen:20 DgmLen:56
Type:3  Code:3  DESTINATION UNREACHABLE: PORT UNREACHABLE
** ORIGINAL DATAGRAM DUMP:
203.166.65.22:1076 -> My.Net.42.15:161 UDP TTL:61 TOS:0x0 ID:0 IpLen:20 DgmLen:69
DF
Len: 49
** END OF DUMP
00 00 00 00 45 00 00 45 00 00 40 00 3D 11 54 09  ....E..E..@.=.T.
CA 57 2A 06 CA 57 2A EA 04 34 00 A1 00 31 E2 DD  .W*..W*..4...1..

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

07/06-12:18:17.333834 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x54
203.166.65.22:1076 -> My.Net.42.15:161 UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:70
DF
Len: 50
30 82 00 26 02 01 00 04 07 70 72 69 76 61 74 65  0..&.....private
A1 18 02 01 01 02 01 00 02 01 00 30 0D 30 82 00  ...........0.0..
09 06 05 2B 06 01 02 01 05 00                    ...+......

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

07/06-12:18:17.334437 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x46
My.Net.42.15 -> 203.166.65.22 ICMP TTL:125 TOS:0x0 ID:19938 IpLen:20 DgmLen:56
Type:3  Code:3  DESTINATION UNREACHABLE: PORT UNREACHABLE
** ORIGINAL DATAGRAM DUMP:
203.166.65.22:1076 -> My.Net.42.15:161 UDP TTL:61 TOS:0x0 ID:0 IpLen:20 DgmLen:70
DF
Len: 50
** END OF DUMP
00 00 00 00 45 00 00 46 00 00 40 00 3D 11 54 08  ....E..F..@.=.T.
CA 57 2A 06 CA 57 2A EA 04 34 00 A1 00 32 28 18  .W*..W*..4...2(.

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

07/06-12:18:20.333817 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x51
203.166.65.22:1076 -> My.Net.42.15:161 UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:67
DF
Len: 47
30 82 00 23 02 01 00 04 04 69 6C 6D 69 A1 18 02  0..#.....ilmi...
01 01 02 01 00 02 01 00 30 0D 30 82 00 09 06 05  ........0.0.....
2B 06 01 02 01 05 00                             +......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/06-12:18:20.334446 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x46
My.Net.42.15 -> 203.166.65.22 ICMP TTL:125 TOS:0x0 ID:19964 IpLen:20 DgmLen:56
Type:3  Code:3  DESTINATION UNREACHABLE: PORT UNREACHABLE
** ORIGINAL DATAGRAM DUMP:
203.166.65.22:1076 -> My.Net.42.15:161 UDP TTL:61 TOS:0x0 ID:0 IpLen:20 DgmLen:67
DF
Len: 47
** END OF DUMP
00 00 00 00 45 00 00 43 00 00 40 00 3D 11 54 0B  ....E..C..@.=.T.
CA 57 2A 06 CA 57 2A EA 04 34 00 A1 00 2F 54 49  .W*..W*..4.../TI

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

07/06-12:18:23.334568 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x51
203.166.65.22:1076 -> My.Net.42.15:161 UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:67
DF
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/06-12:18:17.334437 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x46
My.Net.42.15 -> 203.166.65.22 ICMP TTL:125 TOS:0x0 ID:19938 IpLen:20 DgmLen:56
Type:3  Code:3  DESTINATION UNREACHABLE: PORT UNREACHABLE
** ORIGINAL DATAGRAM DUMP:
203.166.65.22:1076 -> My.Net.42.15:161 UDP TTL:61 TOS:0x0 ID:0 IpLen:20 DgmLen:70
DF
Len: 50
** END OF DUMP
00 00 00 00 45 00 00 46 00 00 40 00 3D 11 54 08  ....E..F..@.=.T.
CA 57 2A 06 CA 57 2A EA 04 34 00 A1 00 32 28 18  .W*..W*..4...2(.

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

[……………………………………SNIP…………………………………….]

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

07/06-12:19:20.331760 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x51
203.166.65.22:1076 -> My.Net.42.15:161 UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:67
DF
Len: 47
30 82 00 23 02 01 00 04 04 72 6D 6F 6E A1 18 02  0..#.....rmon...
01 01 02 01 00 02 01 00 30 0D 30 82 00 09 06 05  ........0.0.....
2B 06 01 02 01 05 00                             +......

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

07/06-12:19:20.332364 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x46
My.Net.42.15 -> 203.166.65.22 ICMP TTL:125 TOS:0x0 ID:20242 IpLen:20 DgmLen:56
Type:3  Code:3  DESTINATION UNREACHABLE: PORT UNREACHABLE
** ORIGINAL DATAGRAM DUMP:
203.166.65.22:1076 -> My.Net.42.15:161 UDP TTL:61 TOS:0x0 ID:0 IpLen:20 DgmLen:67
DF
Len: 47
** END OF DUMP
00 00 00 00 45 00 00 43 00 00 40 00 3D 11 54 0B  ....E..C..@.=.T.
CA 57 2A 06 CA 57 2A EA 04 34 00 A1 00 2F 4E 3E  .W*..W*..4.../N>

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

07/06-12:19:23.333181 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x57
203.166.65.22:1076 -> My.Net.42.15:161 UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:73
DF
Len: 53
30 82 00 29 02 01 00 04 0A 72 6D 6F 6E 5F 61 64  0..).....rmon_ad
```

```
6D 69 6E A1 18 02 01 01 02 01 00 02 01 00 30 0D  min...........0.
30 82 00 09 06 05 2B 06 01 02 01 05 00           0.....+......

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

07/06-12:19:23.333847 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x46
My.Net.42.15 -> 203.166.65.22 ICMP TTL:125 TOS:0x0 ID:20280 IpLen:20 DgmLen:56
Type:3  Code:3  DESTINATION UNREACHABLE: PORT UNREACHABLE
** ORIGINAL DATAGRAM DUMP:
203.166.65.22:1076 -> My.Net.42.15:161 UDP TTL:61 TOS:0x0 ID:0 IpLen:20 DgmLen:73
DF
Len: 53
** END OF DUMP
00 00 00 00 45 00 00 49 00 00 40 00 3D 11 54 05  ....E..I..@.=.T.
CA 57 2A 06 CA 57 2A EA 04 34 00 A1 00 35 0A FF  .W*..W*..4...5..

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

07/06-12:19:26.331836 0:50:BA:ZZ:YY:XX -> 0:B0:C2:ZZ:YY:XX type:0x800 len:0x55
203.166.65.22:1076 -> My.Net.42.15:161 UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:71
DF
Len: 51
30 82 00 27 02 01 00 04 08 68 70 5F 61 64 6D 69  0..'.....hp_admi
6E A1 18 02 01 01 02 01 00 02 01 00 30 0D 30 82  n...........0.0.
00 09 06 05 2B 06 01 02 01 05 00                 ....+......

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

07/06-12:19:26.332505 0:B0:C2:ZZ:YY:XX -> 0:50:BA:ZZ:YY:XX type:0x800 len:0x46
My.Net.42.15 -> 203.166.65.22 ICMP TTL:125 TOS:0x0 ID:20289 IpLen:20 DgmLen:56
Type:3  Code:3  DESTINATION UNREACHABLE: PORT UNREACHABLE
** ORIGINAL DATAGRAM DUMP:
203.166.65.22:1076 -> My.Net.42.15:161 UDP TTL:61 TOS:0x0 ID:0 IpLen:20 DgmLen:71
DF
Len: 51
** END OF DUMP
00 00 00 00 45 00 00 47 00 00 40 00 3D 11 54 07  ....E..G..@.=.T.
CA 57 2A 06 CA 57 2A EA 04 34 00 A1 00 33 78 7E  .W*..W*..4...3x~

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

2.5.1 Source of trace

This detect has been picked up from my network. This day my syslog reported the snmpd service going down a couple of times on one of my hosts (Target Host). This made me turn on packet logging. The packets were captured while the snmpd service was down.

2.5.2 Detect was generated by

The following commands were used to capture and analyze the traffic.

```
# tcpdump -w detect.log -s 65535
```

Explanation:

-w	Writes captured data into a file
-s	Snapshot length. Where default is 68

Complete help on tcpdump is available (`# tcpdump --help`)

The captured log was later analyzed using snort

```
# snort -dev -r detect.log
```

Explanation:

-d	Dumps application layer
-e	Displays second layer header info
-v	Be verbose
-r	Read and process tcpdump file

Complete help on snort is available (`# snort -h`)

2.5.3 Probability of the source address being spoofed

It is unlikely that the source address being spoofed because the attacker wants a response from the target system.

2.5.4 Description of attack

The attacker desires an access into the system information and in some cases take control of the system by cracking the SNMP community string. The attacker can steal system information and statistics if he can crack a string, which has read access. The attacker can also change system configurations if he can crack a community string that has write access to the system.

Generally administrators do not take SNMP very seriously because most don't have good SNMP applications to work with. So SNMP is installed and enabled in the default mode. The attackers take advantage of this. Newer versions of SNMP have lots of built in security features to facilitate better authentication mechanisms and secure the information channels from client to server. Note that this attack has gone undetected by Snort running with its default rule set. This is so because this is a legitimate packet that is been used with malicious intent. If a rule were written to pick up all the packets of this kind then there would be large number of false positives.

2.5.5 Attack mechanism

The attacker is running a scripted tool that tries all the possible default SNMP community strings to gain access into the snmpd server. Such an attack is called a brute force attack. You will find the UDP packets directed to port 161 (SNMPD runs on this port). Scanning through the packets to port 161 you will find that several strings have been tried e.g. public, private, ilmi, rmon, rmon_admin and hp_admin. Surprisingly you will also find an empty packet that has been sent to port 161. Since the service itself is down there is an ICMP Type: 3, Code: 3 (Destination unreachable) message that is sent back to the attacker.

2.5.6 Correlations

The attacker's IP was found in the port scan log, scanning the network for hosts running SNMP. This was noticed a couple of days prior to the incident.

There is a very generic bugtraq that explains that this is a configuration error. The bugtraq is available at http://online.securityfocus.com/bid/973.

2.5.7 Evidence of active targeting

Active targeting was obvious because that was the only host in the network running SNMP and the source address was found in the port scan log. The attacker was performing a network scan looking for hosts running SNMP.

2.5.8 Severity

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

Criticality = 4 (Quite a critical system that also runs SNMPD)
Lethality = 3 (Possible information leak)
System Countermeasures = 4 (SNMP Version 2 was installed and enabled)
Network Countermeasures = 0 (There was no firewall to stop the probe)

Severity = (4 + 3) – (4 + 0)
Severity = 3

2.5.9 Defensive recommendation

In the first case SNMP should only be used where necessary, if the service isn't been used then it should be disabled.

Filtering traffic on SNMP ports using a filtering router or a firewall is one of the best solutions because SNMP is generally used for internal purposes and there is no reason why the world should have access to the service.

SNMP has plenty of vulnerabilities for attacker exploitation. If at all SNMP is necessary for a system, it must be the latest version.

2.5.10 Multiple choice test question

Q. SNMP Community strings are

1. Used to name the community
2. Used to identify the server providing service
3. Are like passwords used for authentication
4. Unique character assigned by RIPE.NET

Answer: 3

SECTION III – Analyze This

3.1 Logs used for analysis

Logs that were used for analysis were recent and were downloaded from http://www.research.umbc.edu/~andy. The analysis covers five days of alerts and port scanning logs viz. from April 12th 2002 to April 17th 2002. The names of the log files can be found in the table below.

Alert Logs	Port Scan Logs	OOS Logs
alert.020412.gz	scans.020412.gz	oos_Apr.12.2002.gz
alert.020413.gz	scans.020413.gz	oos_Apr.13.2002.gz
alert.020414.gz	scans.020414.gz	oos_Apr.14.2002.gz
alert.020415.gz	scans.020415.gz	oos_Apr.15.2002.gz
alert.020416.gz	scans.020416.gz	oos_Apr.16.2002.gz
alert.020417.gz	scans.020417.gz	oos_Apr.17.2002.gz

3.2 Statistical Analysis

This sub section will provide a statistical analysis of the port scan and alert logs. A series of scripts will be run over the logs to provide statistical information for analysis.

NOTE: Both, the alert logs and the port scan logs were merged together into one single file to perform an even analysis over the five days of monitoring. Perl scripts were used to analyze the alert and the port-scan data files. Todd Chapman has written these scripts in his GCIA Practical. Todd Chapman's GCIA Practical can be downloaded from http://www.giac.org/GCIA.php. These scripts have been a boon to all non-programmers..

3.2.1 Alert summary

This gives a tabulated overview of all the alerts (along with statistics) that have been generated in the five days of analysis.

No of Alerts	% of all Alerts	# of distinct SRC IP's	# of distinct DST IP's	Alert Name
407496	39.2	166	5	connect to 515 from inside
152767	14.7	0	0	PORTSCAN DETECTED
84959	8.2	22	17	MISC Large UDP Packet
84403	8.1	405	475	SMB Name Wildcard
75534	7.3	44	19	Watchlist 000220 IL-ISDNNET-990517

73374	7.1	186	868	spp_http_decode: IIS Unicode attack detected
42267	4.1	33	157	SNMP public access
39243	3.8	162	38	ICMP Echo Request L3retriever Ping
18778	1.8	132	127	INFO MSN IM Chat data
13383	1.3	206	181	High port 65535 udp - possible Red Worm - traffic
12047	1.2	43	39	spp_http_decode: CGI Null Byte attack detected
11978	1.2	8843	9	INFO Inbound GNUTella Connect request
6137	0.6	61	4	ICMP Echo Request Nmap or HPING2
2473	0.2	69	80	ICMP Fragment Reassembly Time Exceeded
2462	0.2	19	33	WEB-MISC Attempt to execute cmd
2051	0.2	9	1692	INFO Outbound GNUTella Connect request
1752	0.2	64	2	WEB-IIS view source via translate header
1296	0.1	26	15	FTP DoS ftpd globbing
1210	0.1	15	94	INFO Napster Client Data
1161	0.1	127	1	ICMP Router Selection
500	0.0	41	27	ICMP Echo Request Windows
491	0.0	6	6	Watchlist 000222 NET-NCFC
400	0.0	390	40	IDS552/web-iis_IIS ISAPI Overflow ida nosize
377	0.0	130	3	WEB-IIS _vti_inf access
370	0.0	129	3	WEB-FRONTPAGE _vti_rpc access
346	0.0	1	346	SYN-FIN scan!
283	0.0	7	6	ICMP Destination Unreachable (Protocol Unreachable)
260	0.0	5	25	INFO FTP anonymous FTP
186	0.0	2	2	ICMP Destination Unreachable (Communication Adminis
161	0.0	31	7	ICMP traceroute
150	0.0	45	15	Null scan!
133	0.0	19	25	SCAN Proxy attempt
128	0.0	7	5	SUNRPC highport access!
118	0.0	7	7	High port 65535 tcp - possible Red Worm - traffic
103	0.0	9	19	WEB-MISC 403 Forbidden
93	0.0	28	28	INFO Possible IRC Access
83	0.0	8	2	WEB-CGI scriptalias access
82	0.0	22	11	NMAP TCP ping!
82	0.0	3	3	MISC traceroute
75	0.0	2	1	Tiny Fragments - Possible Hostile Activity
67	0.0	19	15	WEB-MISC compaq nsight directory traversal
53	0.0	8	5	WEB-IIS Unauthorized IP Access Attempt
47	0.0	4	4	WEB-MISC http directory traversal
44	0.0	11	19	INFO - Possible Squid Scan
33	0.0	7	20	Attempted Sun RPC high port access
31	0.0	10	18	Back Orifice

27	0.0	12	14	EXPLOIT x86 NOOP
27	0.0	2	2	ICMP Echo Request BSDtype
25	0.0	11	12	Queso fingerprint
21	0.0	5	6	RFB - Possible WinVNC - 010708-1
20	0.0	20	6	SCAN Synscan Portscan ID 19104
19	0.0	2	3	WEB-IIS encoding access
19	0.0	17	9	EXPLOIT x86 setuid 0
19	0.0	1	1	RPC tcp traffic contains bin_sh
17	0.0	4	4	x86 NOOP - unicode BUFFER OVERFLOW ATTACK
16	0.0	5	1	SCAN FIN
14	0.0	4	4	Port 55850 tcp - Possible myserver activity - ref.
10	0.0	4	4	Port 55850 udp - Possible myserver activity - ref.
9	0.0	2	3	ICMP Echo Request CyberKit 2.2 Windows
9	0.0	4	3	Incomplete Packet Fragments Discarded
8	0.0	7	7	EXPLOIT x86 setgid 0
7	0.0	1	6	FTP MKD / - possible warez site
6	0.0	1	1	WEB-IIS 5 .printer isapi
6	0.0	3	1	WEB-MISC ICQ Webfront HTTP DOS
5	0.0	2	2	INFO Inbound GNUTella Connect accept
5	0.0	1	1	WEB-MISC whisker head
4	0.0	2	2	WEB-CGI formmail access
4	0.0	4	3	EXPLOIT NTPDX buffer overflow
4	0.0	4	1	suspicious host traffic
4	0.0	3	3	TFTP - External TCP connection to internal tftp ser
3	0.0	3	2	TFTP - External UDP connection to internal tftp ser
2	0.0	1	1	TCP SRC and DST outside network
2	0.0	1	2	Probable NMAP fingerprint attempt
2	0.0	1	1	IDS237/web-iis_http-webhits
2	0.0	1	1	WEB-CGI rsh access
2	0.0	1	1	ICMP Destination Unreachable (Host Unreachable)
1	0.0	1	1	MISC PCAnywhere Startup
1	0.0	1	1	WEB-IIS .cnf access
1	0.0	1	1	TFTP - Internal UDP connection to external tftp ser
1	0.0	1	1	FTP EXPLOIT aix overflow
1	0.0	1	1	ICMP SRC and DST outside network
1039790				**TOTAL NO OF ALERTS GENERATED**

SUMMARY OF ALL ALERTS GENERATED

There were 81 unique alerts, the total number of alerts being a high of 1039790 in the five days of activity. The analysis table was generated by processing data. This was generated by a Perl script using Microsoft Excel. This perl script is available in the last section of this document.

3.2.2 Top ten alert sources from the Internal and External network

78.2% of alerts were initiated from the internal network and 21.8% from external network. The table below lists the top ten alert initiating source addresses.

Rank	# of Alerts	SRC IP from Internal Network	Rank	# Of Alerts	SRC IP from External Network
1	193838	MY.NET.153.164	1	56748	61.150.5.19
2	35395	MY.NET.150.83	2	44686	212.179.98.160
3	16274	MY.NET.11.6	3	16344	218.65.86.2
4	15916	MY.NET.11.7	4	10600	212.179.99.10
5	14409	MY.NET.153.136	5	8672	212.179.98.231
6	14188	MY.NET.70.177	6	4601	140.142.8.72
7	11548	MY.NET.153.119	7	4510	212.179.98.254
8	7966	MY.NET.153.211	8	2783	140.142.8.73
9	7785	MY.NET.153.111	9	2585	212.179.99.9
10	6792	MY.NET.153.110	10	2579	209.189.128.78

TOP TEN ALERT SOURCES FROM THE INTERNAL & EXTERNAL NETWORK

3.2.3 Top ten alert destinations in the Internal and External network

This table lists the destination addresses of attacked hosts within the network, which have generated alerts and provides important statistics to an intrusion analyst. The data provides insight into the state of security in the network system. Top ten-destination address in the internal and external network are provided in the table below:

Rank	# of Alerts	DST IP from Internal Network	Rank	# of Alerts	DST IP from External Network
1	371249	MY.NET.150.198	1	4092	209.10.239.135
2	71295	MY.NET.88.162	2	3672	207.189.75.40
3	45097	MY.NET.88.165	3	3193	211.115.213.202
4	35480	MY.NET.11.6	4	2741	211.32.117.26
5	35390	MY.NET.151.77	5	2554	211.115.218.77
6	35024	MY.NET.11.7	6	1978	211.233.29.217
7	17023	MY.NET.152.10	7	1900	207.189.79.124
8	15847	MY.NET.153.146	8	1523	211.233.29.212
9	12159	MY.NET.11.5	9	1385	211.233.28.18
10	9669	MY.NET.150.195	10	1334	211.115.213.207

TOP TEN ALERT DESTINATIONS IN THE INTERNAL & EXTERNAL NETWORK

- Approximately 89.4% out of collected 887023 destination addresses are going to the Internal network.
- In case of the external network, interestingly, 24372 out of the 94016 alerts are generated by the top ten addresses itself.

3.2.4 Top ten destination ports

This statistic indicates the top ten vulnerabilities that attackers can exploit. The destination ports will give clues to the services that run on the ports that possibly are vulnerable to attack. A large percentage of the destination is port 515. Attack on port 515 can disrupt the following services using this port.

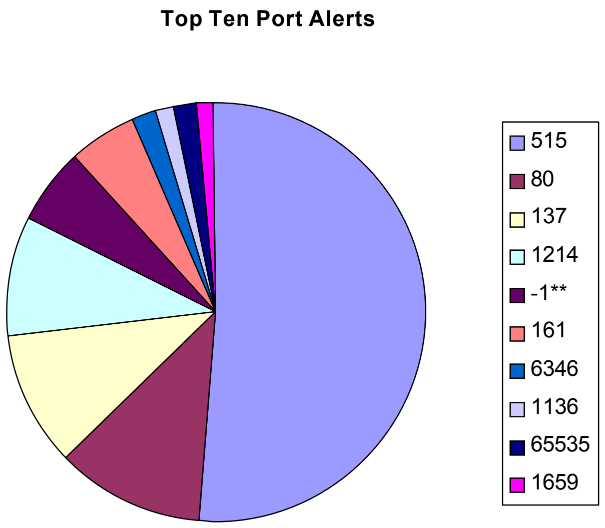
515	tcp	lpdw0rm [trojan] lpdw0rm
515	tcp	printer spooler
515	tcp	Ramen [trojan] Ramen
515	udp	printer spooler

Investigation of the packet source will indicate the intention of the attacker. The top ten ports appearing in the alerts are given below:

Rank	# of Alerts	% of Alerts	Destination Port
1	407496	45.9	515
2	90156	10.2	80
3	84403	9.5	137
4	71425	8.1	1214
5	50258	5.7	-1**
6	42267	4.8	161
7	13717	1.5	6346
8	12970	1.5	1136
9	11644	1.3	65535
10	11599	1.3	1659

TOP TEN DESTINATION PORTS

** Indicates that the alert doesn't have any port address, e.g. ICMP traffic.



GRAPHICAL REPRESENTATION OF THE TOP TEN DESITNATION PORTS

3.2.5 Top ten port scanning sources from the Internal and External network

The top ten source address that have initiated a port scan from within and outside the scanned network have been logged in the port scan files.

INTERNAL NETWROK			EXTERNAL NETWROK		
Rank	**# Of Port Scans**	**SRC IP from Internal Network**	**Rank**	**# Of Port Scans**	**SRC IP from External Network**
1	611241	MY.NET.5.89	1	16000	205.188.228.17
2	517035	MY.NET.60.43	2	15775	205.188.228.1
3	240152	MY.NET.150.143	3	15360	205.188.228.33
4	208858	MY.NET.11.8	4	10746	205.188.228.65
5	198336	MY.NET.6.45	5	9117	205.188.228.129
6	180465	MY.NET.6.48	6	7696	205.188.228.145
7	179465	MY.NET.6.52	7	5584	203.231.232.219
8	141374	MY.NET.6.50	8	4540	209.249.123.160
9	101269	MY.NET.6.51	9	3819	218.65.86.2
10	83358	MY.NET.6.53	10	3740	203.231.232.141

TOP TEN PORT SCANNING SOURCES FROM THE INTERNAL & EXTERNAL NETWORK

3.3 Detailed analysis of the top 10 alerts

Given below are the top ten alerts that were recorded in the five days of monitoring.

Rank	# Of Alerts	% Of all Alerts	Alerts
1	407496	39.2	connect to 515 from inside
2	152767	14.7	PORTSCAN DETECTED
3	84959	8.2	MISC Large UDP Packet
4	84403	8.1	SMB Name Wildcard
5	75534	7.3	Watchlist 000220 IL-ISDNNET-990517
6	73374	7.1	spp_http_decode: IIS Unicode attack detected
7	42267	4.1	SNMP public access
8	39243	3.8	ICMP Echo Request L3retriever Ping
9	18778	1.8	INFO MSN IM Chat data
10	13383	1.3	High port 65535 udp - possible Red Worm - traffic

Analysis of these alerts will create an understanding of whose who of the network. In this study, the first five alerts will be analyzed in depth while the remaining five only up to a point of understanding the alert and statistics that show active targeting.

3.3.1 Analysis of 'Connect to 515 from inside'

3.3.1.1 Alert samples

```
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515
[**] connect to 515 from inside [**] MY.NET.150.83:512 -> MY.NET.151.77:515

 [...............................................SNIP..............................................]

[**] connect to 515 from inside [**] MY.NET.153.168:2073 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.153.168:2073 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.153.168:2073 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.153.168:2073 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.88.151:1218 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.88.151:1218 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.88.151:1218 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.88.151:1218 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.88.151:1218 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.88.151:1218 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.88.151:1218 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.88.151:1218 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.88.151:1218 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.88.151:1218 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.88.151:1218 -> MY.NET.150.198:515
```

```
[**] connect to 515 from inside [**] MY.NET.88.151:1218 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.88.151:1218 -> MY.NET.150.198:515
[**] connect to 515 from inside [**] MY.NET.88.151:1218 -> MY.NET.150.198:515
```

Note: The timestamp has been erased from the log entries above for compact viewing.

First alert captured at: 04/12-00:00:03.277938
Last alert captured at: 04/17-23:51:47.138107

3.3.1.2 Rule

A custom customized rule written to alert all the internal connections made to port 515.

3.3.1.3 Source and destination addresses involved

The alert data created by the source to destination address and the number of connects to port 515 from inside would form the body of data for analysis. A Perl script would parse the alerts' file and generate the output. The Perl script is written by Todd Chapman in his GCIA practical, available http://www.giac.org/GCIA.php as well as in the last subsection of this paper.

```
# perl alert_stats.pl '515 from inside' alert

MY.NET.152.126  -> MY.NET.150.198        alerts: 14
MY.NET.153.105  -> MY.NET.150.198        alerts: 4050
MY.NET.153.106  -> MY.NET.150.198        alerts: 2317
MY.NET.153.107  -> MY.NET.150.198        alerts: 2912
MY.NET.153.45   -> MY.NET.150.198        alerts: 269
MY.NET.153.108  -> MY.NET.150.198        alerts: 1795
MY.NET.153.46   -> MY.NET.150.198        alerts: 189
MY.NET.153.109  -> MY.NET.150.198        alerts: 1678
MY.NET.153.180  -> MY.NET.150.198        alerts: 278
MY.NET.153.182  -> MY.NET.150.198        alerts: 47
MY.NET.153.184  -> MY.NET.150.198        alerts: 66
MY.NET.153.185  -> MY.NET.150.198        alerts: 17
MY.NET.149.23   -> MY.NET.1.63            alerts: 18
MY.NET.153.186  -> MY.NET.150.198        alerts: 409
MY.NET.153.187  -> MY.NET.150.198        alerts: 374
MY.NET.153.188  -> MY.NET.150.198        alerts: 129
MY.NET.149.27   -> MY.NET.1.63            alerts: 50
MY.NET.153.189  -> MY.NET.150.198        alerts: 402
MY.NET.149.28   -> MY.NET.1.63            alerts: 19
MY.NET.114.73   -> MY.NET.150.198        alerts: 251
MY.NET.149.29   -> MY.NET.1.63            alerts: 30
MY.NET.153.110  -> MY.NET.150.198        alerts: 2609
MY.NET.153.111  -> MY.NET.150.198        alerts: 6541
MY.NET.153.112  -> MY.NET.150.198        alerts: 1069
MY.NET.153.113  -> MY.NET.150.198        alerts: 5485
MY.NET.153.114  -> MY.NET.150.198        alerts: 4095
MY.NET.153.115  -> MY.NET.150.198        alerts: 4322
MY.NET.153.117  -> MY.NET.150.198        alerts: 3697
MY.NET.153.118  -> MY.NET.150.198        alerts: 3702
MY.NET.153.119  -> MY.NET.150.198        alerts: 11466
MY.NET.153.190  -> MY.NET.150.198        alerts: 306
MY.NET.153.193  -> MY.NET.150.198        alerts: 727
MY.NET.153.194  -> MY.NET.150.198        alerts: 126
MY.NET.153.195  -> MY.NET.150.198        alerts: 3036
MY.NET.153.196  -> MY.NET.150.198        alerts: 1101
MY.NET.153.197  -> MY.NET.150.198        alerts: 308
MY.NET.153.198  -> MY.NET.150.198       alerts: 1172
MY.NET.153.199  -> MY.NET.150.198       alerts: 295
MY.NET.150.83   -> MY.NET.151.77        alerts: 35359
MY.NET.152.44   -> MY.NET.150.198       alerts: 683
MY.NET.152.213  -> MY.NET.150.198       alerts: 167
MY.NET.152.214  -> MY.NET.150.198       alerts: 176
MY.NET.152.45   -> MY.NET.150.198       alerts: 199
MY.NET.153.120  -> MY.NET.150.198       alerts: 2663
MY.NET.153.121  -> MY.NET.150.198       alerts: 1734
MY.NET.152.215  -> MY.NET.150.198       alerts: 484
MY.NET.152.46   -> MY.NET.150.198       alerts: 460
MY.NET.152.216  -> MY.NET.150.198       alerts: 294
MY.NET.153.123  -> MY.NET.150.198       alerts: 4835
MY.NET.153.124  -> MY.NET.150.198       alerts: 4030
MY.NET.153.125  -> MY.NET.150.198       alerts: 3478
MY.NET.153.126  -> MY.NET.150.198       alerts: 4624
MY.NET.153.127  -> MY.NET.150.198       alerts: 1703
MY.NET.153.202  -> MY.NET.150.198       alerts: 1229
MY.NET.153.203  -> MY.NET.150.198       alerts: 416
MY.NET.153.205  -> MY.NET.150.198       alerts: 1495
MY.NET.153.206  -> MY.NET.150.198       alerts: 694
MY.NET.153.71   -> MY.NET.150.198       alerts: 730
MY.NET.153.208  -> MY.NET.150.198       alerts: 32
MY.NET.153.135  -> MY.NET.150.198       alerts: 2946
MY.NET.152.157  -> MY.NET.150.198       alerts: 374
MY.NET.153.136  -> MY.NET.150.198       alerts: 12974
MY.NET.153.209  -> MY.NET.150.198       alerts: 1077
MY.NET.153.137  -> MY.NET.150.198       alerts: 1169
MY.NET.152.158  -> MY.NET.150.198       alerts: 441
MY.NET.152.159  -> MY.NET.150.198       alerts: 464
MY.NET.149.51   -> MY.NET.1.63           alerts: 41
MY.NET.88.148   -> MY.NET.150.198       alerts: 5260
MY.NET.153.210  -> MY.NET.150.198       alerts: 16
MY.NET.153.211  -> MY.NET.150.198       alerts: 6023
MY.NET.152.160  -> MY.NET.150.198       alerts: 460
MY.NET.153.140  -> MY.NET.150.198       alerts: 1253
[................................SNIP................................]
MY.NET.152.183  -> MY.NET.150.198       alerts: 1435
MY.NET.152.184  -> MY.NET.150.198       alerts: 431
MY.NET.153.163  -> MY.NET.150.198       alerts: 1013
MY.NET.152.185  -> MY.NET.150.198       alerts: 107
MY.NET.153.164  -> MY.NET.150.198       alerts: 193715
MY.NET.153.165  -> MY.NET.150.198       alerts: 50
MY.NET.152.186  -> MY.NET.150.198       alerts: 127
MY.NET.153.166  -> MY.NET.150.198       alerts: 74
MY.NET.153.167  -> MY.NET.150.198       alerts: 569
MY.NET.153.168  -> MY.NET.150.198       alerts: 229
[................................SNIP................................]
MY.NET.153.173  -> MY.NET.150.198       alerts: 664
MY.NET.153.174  -> MY.NET.150.198       alerts: 2203
MY.NET.153.175  -> MY.NET.150.198       alerts: 21
MY.NET.153.176  -> MY.NET.150.198       alerts: 309
MY.NET.153.177  -> MY.NET.150.198       alerts: 630
MY.NET.153.179  -> MY.NET.150.198       alerts: 565
MY.NET.152.20   -> MY.NET.150.198       alerts: 998
MY.NET.152.21   -> MY.NET.150.198       alerts: 254
MY.NET.152.22   -> MY.NET.150.198       alerts: 274
```

3.3.1.4 Description and analysis

The applications that run on port 515 are:

515	tcp	lpdw0rm [trojan] lpdw0rm
515	tcp	Ramen [trojan] Ramen

515	udp	printer spooler
515	tcp	printer spooler

One could safely assume that most networks are free from "Trojan" or some other worm infection. Todd Chapman studied this network for his GCIA practical and did not find 'connect to 515 from inside' in the top five alerts. Practical sense also would indicate that a worthy network couldn't stay under virus attack for prolonged period of 4 months.

- This alert is a TCP connection triggered. UNIX LPD printer spooler services when requested would be looking for internal hosts trying to connect to TCP port 515. The UNIX LPD printer spooler has several vulnerabilities and is easily exploitable.

Some references on lprng overflow are given in the correlations sections below.

3.3.1.5 Signs of active targeting

Chances of active targeting is low because the ID rule picks up all the addresses from the internal network that make a connection to the lpd service.

Majority of the addresses may have been false positives because users from the internal network would genuinely be seeking connections to the lpd service.

One can still study the top three addresses giving alerts to see if they had any malicious intent.

```
MY.NET.150.83   -> MY.NET.151.77        alerts: 35359
```

This address is second amongst the top ten internal source address-generating alerts. Noteworthy here, is the fact that the source has also generated 35359 alerts. This leads us to believe that this particular host is probably not linked with any other alert other than this. This address may have a forwarder to the lpd service running on MY.NET.151.77.

```
MY.NET.153.136  -> MY.NET.150.198       alerts: 12974
```

This host is involved as source in 14409 alerts. To find out what were the other alerts the host was involved with we ran the following command

```
# grep "MY.NET.153.136.*->" alert | cut -b 23-60 | sort | uniq -c
  12974  [**] connect to 515 from inside [**]
      8  [**] ICMP Router Selection [**] MY.NE
     63  [**] INFO MSN IM Chat data [**] MY.NE
   1364  [**] spp_http_decode: IIS Unicode att
```

This host may have been infected with a worm, as it is also the source for the IIS Unicode attack that has generated 1364 alerts.

- The source should be checked and scanned for infections
- If its running IIS then the latest service packs and patches should be applied. These patches are available at http://www.microsoft.com/technet.

Although the lpd connections seem genuine, one needs to find out more on the particular host.

```
MY.NET.153.164  -> MY.NET.150.198        alerts: 193715
```

To find out the alerts other than the 1,93,715 alerts the host was involved in, the following command was run:

```
# grep "MY.NET.153.164.*->" alert | cut -b 23-60 | sort | uniq -c
 193715  [**] connect to 515 from inside [**]
      3  [**] High port 65535 udp - possible R
     26  [**] ICMP Fragment Reassembly Time Ex
     94  [**] spp_http_decode: IIS Unicode att
```

Even this is a source for an IIS Unicode attack.

- The source should be checked and scanned for infections
- If its running IIS then the latest available service packs and patches should be applied. These patches are available at http://www.microsoft.com/technet.

Now that we have studied the sources, the focus must shift to study the top three destination addresses.

```
MY.NET.150.198
```

The following command was run to find out all the alerts with MY.NET.150.198 being the destination address.

```
# grep "> MY.NET.150.198" alert | cut -b 23-60 | sort | uniq -c
 371244  [**] connect to 515 from inside [**]
      3  [**] ICMP Echo Request Windows [**] M
      1  [**] SMB Name Wildcard [**] MY.NET.22
      1  [**] SYN-FIN scan! [**] 209.66.74.90:
```

In most cases, the alert is a false positive. The address in question in most cases would be an lpd server serving genuine lpd requests. Alerts other than the 515 connect are few.

```
MY.NET.1.63
```

On running the command below, the host in question has generated 212 alerts, all of which are 'connect to 515 from inside'. Again, this is a case of a false positive, where a genuine lpd server has generated alerts.

```
# grep "> MY.NET.1.63" alert | cut -b 23-60 | sort | uniq -c
    212  [**] connect to 515 from inside [**]
```

```
MY.NET.151.77
```

To find out all the alerts where the above address is the destination address we ran the following command.

```
# grep "> MY.NET.151.77" alert | cut -b 23-60 | sort | uniq -c
  35359  [**] connect to 515 from inside [**]
      2  [**] IDS552/web-iis_IIS ISAPI Overflo
```

```
     2  [**] SMB Name Wildcard [**] MY.NET.22
    27  [**] SNMP public access [**] MY.NET.1
```

From the output, it is evident that majority of alerts are 'connect to 515 from inside' rule, all case of a false positive. Interestingly there is one more alert that tells us that the host has been accessed using SNMP using the default 'public' community sting.

- The community strings of the host must be checked and replaced by non-default values after confirmation. One must ensure that the SNMP software running on the host is updated.

3.3.1.6 Defensive recommendation

Confirmation of the presence of an lprng worm in the hosts as indicated in the alert destinations must be done and remedial measures taken. Detailed description, updates and patches for the vulnerability are available at http://online.securityfocus.com/bid/1712/solution/

3.3.1.7 Correlations

A proper description of the alert with references is available at the following links
http://icat.nist.gov/icat.cfm?cvename=CVE-2000-0917
http://www.iss.net/security_center/static/5287.php
http://online.securityfocus.com/bid/1712

3.3.2 Analysis of 'PORTSCAN DETECTED'

3.3.2.1 Alert samples

```
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.152.176 (THRESHOLD 4 connections
exceeded in 0 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.153.154 (THRESHOLD 4 connections
exceeded in 0 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.60.43 (THRESHOLD 4 connections
exceeded in 5 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.153.112 (THRESHOLD 4 connections
exceeded in 2 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.153.117 (THRESHOLD 4 connections
exceeded in 2 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.11.8 (THRESHOLD 4 connections
exceeded in 6 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.152.20 (THRESHOLD 4 connections
exceeded in 2 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.6.60 (THRESHOLD 4 connections
exceeded in 0 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.153.168 (THRESHOLD 4 connections
exceeded in 1 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.152.170 (THRESHOLD 4 connections
exceeded in 0 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.152.180 (THRESHOLD 4 connections
exceeded in 2 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.60.43 (THRESHOLD 4 connections
exceeded in 1 seconds) [**]
```

```
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.152.185 (THRESHOLD 4 connections
exceeded in 2 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.152.174 (THRESHOLD 4 connections
exceeded in 0 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.88.176 (THRESHOLD 4 connections
exceeded in 9 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.153.127 (THRESHOLD 4 connections
exceeded in 1 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.153.112 (THRESHOLD 4 connections
exceeded in 1 seconds) [**]

[………………………………………SNIP………………………………………]

[**] spp_portscan: PORTSCAN DETECTED from 211.14.5.76 (THRESHOLD 4 connections
exceeded in 31 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.150.143 (THRESHOLD 4 connections
exceeded in 2 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.88.151 (THRESHOLD 4 connections
exceeded in 4 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.60.43 (THRESHOLD 4 connections
exceeded in 2 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.152.168 (THRESHOLD 4 connections
exceeded in 6 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.5.89 (THRESHOLD 4 connections
exceeded in 1 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.11.8 (THRESHOLD 4 connections
exceeded in 7 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.152.44 (THRESHOLD 4 connections
exceeded in 0 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.11.7 (THRESHOLD 4 connections
exceeded in 3 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.152.165 (THRESHOLD 4 connections
exceeded in 0 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.1.52 (THRESHOLD 4 connections
exceeded in 0 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.152.45 (THRESHOLD 4 connections
exceeded in 0 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.11.8 (THRESHOLD 4 connections
exceeded in 9 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.6.60 (THRESHOLD 4 connections
exceeded in 10 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.5.89 (THRESHOLD 4 connections
exceeded in 0 seconds) [**]
[**] spp_portscan: PORTSCAN DETECTED from MY.NET.152.183 (THRESHOLD 4 connections
exceeded in 0 seconds) [**]
```

Note: The timestamp has been erased from the log entries above for compact viewing.

First alert captured at: 04/12-00:16:01.304655
Last alert captured at: 04/18-00:06:32.155979

3.3.2.2 Rule

There is a complete rules file for port scans. The port scan logs will indicate the specific scenario out of the several possibilities of generating alerts. A sample rule for a port scan is given below.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN synscan portscan"; id:
39426; flags: SF;reference:arachnids,441; classtype:attempted-recon; sid:630;
rev:1;)
```

3.3.2.3 Source addresses involved

Normally a portscan detect is large in number. As such 'PORTSCAN DETECTED' alerts are designed to list only the top thirty hosts involved.

```
RANK   ALERTS        HOSTS
1      5558          MY.NET.150.143
2      3737          MY.NET.11.8
3      3000          MY.NET.5.89
4      2826          MY.NET.6.52
5      2797          MY.NET.60.43
6      2714          MY.NET.11.7
7      2641          MY.NET.6.45
8      2530          MY.NET.11.6
9      2468          MY.NET.6.48
10     2453          MY.NET.6.50
11     2162          MY.NET.1.52
12     2131          MY.NET.6.60
13     2001          MY.NET.6.53
14     1796          MY.NET.6.51
15     1449          MY.NET.152.19
16     1246          MY.NET.60.11
17     1127          MY.NET.153.211
18     1095          MY.NET.153.157
19     1082          MY.NET.152.157
20     1001          MY.NET.153.146
21     991           MY.NET.152.165
22     982           MY.NET.152.164
23     972           MY.NET.152.15
24     968           MY.NET.153.196
25     959           MY.NET.150.246
26     939           MY.NET.152.183
27     914           MY.NET.153.173
28     901           MY.NET.152.22
29     893           MY.NET.152.248
30     890           MY.NET.152.163
```

It is important to note that there are no foreign addresses presenting the list. No conclusions should be drawn yet; one must first analyze the top two hosts that have generated port-scanning alerts.

3.3.2.4 Description and analysis

'PORTSCAN DETECED' is a generic alert. To understand the attack trend, analysis of the top two-source address generating the highest number of portscan alerts must be done. It is important to understand the nature of portscans that are initiated.

Spoofed addresses

In an advanced port scanning mechanism the attacker can use a dummy host to scan the target hosts. This advanced technique is called the TCP idle scan. This scan uses spoofed source address.

More info on the TCP Idle scan technique can be read at the link http://www.hping.org/papers.html.

3.3.2.5 Signs of active targeting

The address list of the 'portscan detected' error is voluminous, only the top two sources from the list of port scan alerts will be studied.

MY.NET.150.143

The host MY.NET.150.143 has generated 5558 portscan-detected alerts. To analyze the source address, the command below was run:

```
# grep "MY.NET.150.143.* ->" scans | cut -b 40- | uniq -c
      1 202.202.202.202:86 SYN ******S*
      1 62.163.118.244:4665 UDP
      1 137.226.239.163:4665 UDP
      1 80.13.60.153:4665 UDP
      1 216.63.109.137:7665 UDP
      1 129.241.150.180:4665 UDP
      1 194.192.25.235:4665 UDP
      1 213.70.139.26:8665 UDP
      1 213.139.173.239:4665 UDP
      1 194.192.25.235:4665 UDP
      1 213.70.139.26:8665 UDP
      1 202.202.202.202:86 SYN ******S*
      1 137.226.239.163:4665 UDP
      1 62.163.118.244:4665 UDP
      1 200.83.55.167:4662 SYN ******S*
      1 216.63.109.137:7665 UDP
      1 140.113.27.100:4665 UDP
      1 210.243.45.194:4662 SYN ******S*
      2 140.113.27.100:4665 UDP
      1 MY.NET.150.1:1900 UDP
      1 211.23.154.5:4662 SYN ******S*
      1 140.113.27.100:4665 UDP
      1 211.23.154.5:4662 SYN ******S*
      2 202.202.202.202:86 SYN ******S*
[………………………………………SNIP………………………………………]
      1 18.250.4.136:4662 SYN ******S*
      1 211.74.160.243:4662 SYN ******S*
      1 203.73.128.217:4662 SYN ******S*
      1 203.204.89.244:4662 SYN ******S*
      1 140.131.31.85:4662 SYN ******S*
      1 202.202.202.202:86 SYN ******S*
      1 140.131.31.85:4662 SYN ******S*
      1 MY.NET.150.1:2234 UDP
      2 61.220.235.9:4662 SYN ******S*
      1 202.108.35.19:80 SYN ******S*
      1 MY.NET.150.1:1900 UDP
      2 211.23.128.98:4662 SYN ******S*
      1 61.56.206.81:4662 SYN ******S*
      1 202.145.66.252:4662 SYN ******S*
      1 202.202.202.202:86 SYN ******S*
```

The scans are random, where there are no two addresses scanned in a single network. The ports that are scanned are also random. For example port 86, 1900, 4665, 8664, 7665 etc. There could be a problem with the command output because apparently, there is profiling of random hosts on random networks.

Involvement of the host in other alerts must be found by running the command below: Subsequently, a command is run to find out if the host has been a destination address to any of the alerts to understand if the host was exploited in any way.

```
# grep "MY.NET.150.143.*->" alert | cut -b 23-43 | sort | uniq -c
      8  [**] High port 65535
     25  [**] INFO MSN IM Cha
      2  [**] SMB Name Wildca
     17  [**] spp_http_decode
      2  [**] WEB-IIS Unautho
      1  [**] WEB-MISC 403 Fo
```

This host has been involved in several alerts that have been picked for analysis. The command below will confirm the involvement of the host as a destination in an alert.

```
# grep "> MY.NET.150.143" alert | cut -b 23-43 | sort | uniq -c
      8  [**] EXPLOIT x86 set
      7  [**] High port 65535
     19  [**] IDS552/web-iis_
      4  [**] INFO FTP anonym
     24  [**] INFO MSN IM Cha
      2  [**] Null scan! [**]
      4  [**] Queso fingerpri
      1  [**] SCAN Synscan Po
      2  [**] SMB Name Wildca
    312  [**] spp_http_decode
      1  [**] SYN-FIN scan! [
    337  [**] Watchlist 00022
    256  [**] WEB-MISC Attemp
```

The first alert on the list above is 'EXPLOIT x86 setuid 0'. It was found that all the source addresses were foreign address and with a solitary host that was being exploited. This is the host under question.

The system is prone to many high priority attacks and the chances of the system infection are high.

MY.NET.11.8

This interesting host specimen has been scanning hosts within the networks. The command below was run to confirm this.

```
# grep "MY.NET.11.8.* ->" scans | cut -b 40-47 | uniq -c
 208858 NET.152.
```

To find out what scans the host has been conducting the command below was run. The output is interesting.

```
# grep "MY.NET.11.8.* ->" scans | cut -b 40- | uniq -c
      1 NET.152.176:1346 UDP
      1 NET.152.13:1346 UDP
      1 NET.152.178:1346 UDP
      1 NET.152.163:1346 UDP
      1 NET.152.250:1346 UDP
      1 NET.152.181:1346 UDP
      1 NET.152.175:1346 UDP
      1 NET.152.184:1346 UDP
      1 NET.152.172:1346 UDP
      1 NET.152.18:1346 UDP
      1 NET.152.159:1346 UDP
      1 NET.152.251:1346 UDP
      1 NET.152.176:1346 UDP
      1 NET.152.13:1346 UDP
      1 NET.152.250:1346 UDP
      1 NET.152.181:1346 UDP
      1 NET.152.175:1346 UDP
      1 NET.152.247:1346 UDP
      1 NET.152.184:1346 UDP
      1 NET.152.172:1346 UDP
      1 NET.152.18:1346 UDP
      1 NET.152.159:1346 UDP
      1 NET.152.251:1346 UDP
      1 NET.152.176:1346 UDP
      1 NET.152.13:1346 UDP
      1 NET.152.178:1346 UDP
      1 NET.152.163:1346 UDP
      1 NET.152.250:1346 UDP
      1 NET.152.181:1346 UDP
      1 NET.152.175:1346 UDP
      1 NET.152.247:1346 UDP
      1 NET.152.184:1346 UDP
      1 NET.152.172:1346 UDP
      1 NET.152.18:1346 UDP
      1 NET.152.159:1346 UDP
      1 NET.152.251:1346 UDP
      1 NET.152.176:1346 UDP
      1 NET.152.13:1346 UDP
      1 NET.152.178:1346 UDP
      1 NET.152.163:1346 UDP
      1 NET.152.250:1346 UDP
      1 NET.152.181:1346 UDP
      1 NET.152.175:1346 UDP
      1 NET.152.247:1346 UDP
      1 NET.152.184:1346 UDP
      1 NET.152.172:1346 UDP
      1 NET.152.18:1346 UDP
      1 NET.152.159:1346 UDP
      1 NET.152.251:1346 UDP
      1 NET.152.176:1346 UDP
      1 NET.152.13:1346 UDP
[………………………………………SNIP…………………………………………]
```

The host is desperately looking for a responsive host. The output above indicates that it is scanning the whole network to find any host that responds to UDP port 1346. This port is used by the Symantec Ghost multicast.(client).

This host is likely to be a Symantec Ghost multicast server that would be scanning the entire network to locate hosts that have the Ghost client installed. The scan seems very specific and doesn't look malicious.

3.3.2.6 Defensive recommendation

This host MY.NET.150.143 needs to be checked for any worm or a random scanner that may be installed on the system and for Trojan virus infection of the system.

The second host should be checked for server application that is scanning the network to locate responsive clients ready to connect.

3.3.2.7 Correlations

Documents on portscans are available on the Internet. For more information, visit http://www.insecure.org; http://it.sans.org; http://www.cert.org

3.3.3 Analysis of 'MISC Large UDP Packet'

3.3.3.1 Alert samples

```
[**] MISC Large UDP Packet [**] 211.233.70.161:0 -> MY.NET.153.143:0
[**] MISC Large UDP Packet [**] 211.233.70.161:0 -> MY.NET.153.143:0
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354

[..............................................SNIP..............................................]

[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:0 -> MY.NET.153.143:0
[**] MISC Large UDP Packet [**] 211.233.70.161:3032 -> MY.NET.153.143:1354
[**] MISC Large UDP Packet [**] 211.233.70.161:7000 -> MY.NET.153.143:7001
[**] MISC Large UDP Packet [**] 211.98.95.30:0 -> MY.NET.153.210:0
[**] MISC Large UDP Packet [**] 211.98.95.30:4209 -> MY.NET.153.210:1193
[**] MISC Large UDP Packet [**] 211.98.95.30:4209 -> MY.NET.153.210:1193
[**] MISC Large UDP Packet [**] 211.98.95.30:4209 -> MY.NET.153.210:1193
```

Note: The timestamp has been erased from the log entries above for compact viewing.

First alert captured at: 04/12-12:35:42.827839
Last alert captured at: 04/12-15:04:33.399702

3.3.3.2 Rule

```
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;)
```

3.3.3.3 Source and destinations involved

```
# perl alert_stats.pl 'MISC Large UDP Packet' alert
207.25.79.240   -> MY.NET.152.250       alerts: 1
202.57.96.18    -> MY.NET.152.177       alerts: 350
211.98.95.30    -> MY.NET.153.210       alerts: 505
61.150.5.19     -> MY.NET.153.146       alerts: 7919
61.150.5.19     -> MY.NET.88.137        alerts: 1206
61.150.5.19     -> MY.NET.88.165        alerts: 45084
61.150.5.19     -> MY.NET.153.159       alerts: 2538
140.142.8.72    -> MY.NET.153.146       alerts: 4601
66.77.13.123    -> MY.NET.151.127       alerts: 189
140.142.8.73    -> MY.NET.153.146       alerts: 2783
207.46.235.140  -> MY.NET.153.45        alerts: 65
216.106.172.150 -> MY.NET.153.145       alerts: 1525
210.181.4.200   -> MY.NET.152.46        alerts: 53
63.240.15.204   -> MY.NET.153.159       alerts: 576
216.106.172.144 -> MY.NET.153.145       alerts: 492
63.240.15.207   -> MY.NET.153.159       alerts: 108
202.103.30.118  -> MY.NET.88.140        alerts: 71
211.233.70.161  -> MY.NET.153.143       alerts: 19
211.233.70.163  -> MY.NET.153.174       alerts: 167
128.2.203.61    -> MY.NET.153.170       alerts: 4
211.233.25.31   -> MY.NET.153.174       alerts: 102
218.65.86.2     -> MY.NET.152.10        alerts: 16338
211.233.25.27   -> MY.NET.153.174       alerts: 66
64.157.3.151    -> MY.NET.153.197       alerts: 38
63.240.15.199   -> MY.NET.153.159       alerts: 159
```

The above table was generated using a perl script called alert_stats.pl, the source of which can be found in the last section of this paper.

3.3.3.4 Description and analysis

In the, output all the sources generating the alert are external addresses while all the destination hosts are internal addresses. This snort rule picks up all the UDP packets larger than 4000 bytes. A few possible applications generating such kind of traffic can be thought of.

Some voice conferencing software's do use UDP that can sometimes exceed 4000 bytes. If such an application is generating the alerts. Streaming audio sites also use UDP to stream channels to client hosts. As such, one has to further analyze the alerts and see if the intent is malicious.

3.3.3.5 Signs of active targeting

As a sample, the top two source and destination addresses that have generated this alert will be analyzed.

```
61.150.5.19     -> MY.NET.88.165         alerts: 45084
```

It is important to know if the destination address has appeared as the destination address in other alerts as well. With this objective, the script below was run.

```
# grep "> MY.NET.88.165" alert | cut -b 23-43 | sort | uniq -c
      1  [**] High port 65535
      3  [**] INFO - Possible
  45084  [**] MISC Large UDP
      5  [**] SCAN Proxy atte
      4  [**] SMB Name Wildca
```

The output indicates that this address has received a 45084 large UDP packet. It is important to find out if this source address has appeared as source in more then this alert by running the script below:

```
# grep "61.150.5.19.*->" alert | cut -b 23-43 | sort | uniq -c
      1  [**] High port 65535
  56747  [**] MISC Large UDP
```

The output is innocuous. A whois is done on the system to origins of the packets.

```
# whois 61.150.5.19@whois.apnic.net
[whois.apnic.net]

inetnum:      61.150.0.0 - 61.150.31.255
netname:      SNXIAN
descr:        xi'an data branch,XIAN CITY SHAANXI PROVINCE
country:      CN
admin-c:      WWN1-AP
tech-c:       WWN1-AP
mnt-by:       MAINT-CHINANET-SHAANXI
mnt-lower:    MAINT-CN-SNXIAN
changed:      ipadm@public.xa.sn.cn 20010309
source:       APNIC

person:       WANG WEI NA
address:      Xi Xin street 90# XIAN
country:      CN
phone:        +8629-724-1554
fax-no:       +8629-324-4305
e-mail:       xaipadm@public.xa.sn.cn
nic-hdl:      WWN1-AP
mnt-by:       MAINT-CN-SNXIAN
changed:      wwn@public.xa.sn.cn 20001127
source:       APNIC
```

Apparently, a dialup account had sent UDP packets to this particular host. These packets could be spurious packets with malicious intent. For lack of evidence, one cannot jump to this conclusion, but cannot rule out the possibility.

- The host should be watched carefully for intrusion and if detected, corrective measures should be taken.

Last of all, one must determine if the destination address is involved in initiating the alert.

```
# grep "MY.NET.88.165.*->" alert | cut -b 23-43 | sort | uniq -c
    548  [**] ICMP Fragment R
      2  [**] SMB Name Wildca
     31  [**] spp_http_decode
```

Although the analysis as yet has not revealed startling situations, the probing of the destination host to find traces of intrusion on the system is necessary. If there are no traces detected, one could conclude that some voice conferencing software is running and that the large UDP packets for communication is being sent by the application.

```
218.65.86.2     -> MY.NET.152.10         alerts: 16338
```

Taking a similar approach, one must find out if the source address is involved in initiating any other alert. With this as objective, the command below is run:

```
# grep "> MY.NET.152.10" alert | cut -b 23-43 | sort | uniq -c
     68  [**] High port 65535
  16338  [**] MISC Large UDP
    617  [**] SMB Name Wildca
   7325  [**] SNMP public acc
```

The output clearly shows that the host has been accessed using default SNMP community strings apart from the 16338 alerts generated by large UDP packets. No other conclusions can be drawn. Now the source of alerts must be checked.

```
# grep "218.65.86.2.*->" alert | cut -b 23-43 | sort | uniq -c
      6  [**] High port 65535
  16338  [**] MISC Large UDP
```

The output patterns are similar to the last one and no conclusions can be made at first sight. One must run whois to locate whom the address belongs, and see if some clue can be found.

```
# whois 218.65.86.2@whois.apnic.net
[whois.apnic.net]

% How to use the APNIC Whois Database   www.apnic.net/db/
% Upgrade to Whois v3 on 20 August 2002 www.apnic.net/whois-v3
% Whois data copyright terms            www.apnic.net/db/dbcopyright.html

inetnum:     218.64.0.0 - 218.65.127.255
netname:     CHINANET-JX
descr:       CHINANET jiangxi province network
descr:       China Telecom
descr:       A12,Xin-Jie-Kou-Wai Street
descr:       Beijing 100088
country:     CN
admin-c:     CH93-AP
tech-c:      WZ1-CN
mnt-by:      MAINT-CHINANET
mnt-lower:   MAINT-IP-WWF
changed:     hostmaster@ns.chinanet.cn.net 20010719
source:      APNIC
```

```
person:      Chinanet Hostmaster
address:     No.31 ,jingrong street,beijing
address:     100032
country:     CN
phone:       +86-10-66027112
fax-no:      +86-10-66027334
e-mail:      hostmaster@ns.chinanet.cn.net
nic-hdl:     CH93-AP
mnt-by:      MAINT-CHINANET
changed:     shenjun@cndata.com 20020627
source:      APNIC

person:      Wanshu Zhou
address:     Data Communication Bureau  MPT
address:     40 Xueyuan  Rd.
address:     Beijing  China  100083
phone:       +86-10-205-3992
fax-no:      +86-10-205-3994
e-mail:      zhouws@public.bta.net.cn
nic-hdl:     WZ1-CN
notify:      zhouws@public.bta.net.cn
notify:      zhang@usai.asiainfo.com
mnt-by:      MAINT-NULL
changed:     zhang@usai.asiainfo.com 19960115
source:      APNIC
```

Whois output indicates that the source originates from China. There is no permanent host that holds this address. As such, this address could belong to an ISP. As such, there is a possibility of it being a dialup address. This is very similar to the analysis done earlier.

Probably this might be the voice conferencing software that uses UDP unless further investigations conclude otherwise.

3.3.3.6 Defensive recommendation

The recommendation for the cases we have seen is as follows.

- Investigations must continue to identify intrusion signatures in both the cases.
- The second host must upgrade the installed SNMP application and ensure that the default community strings is changed to non- default values.

3.3.3.7 Correlations

Although correlation of this kind of traffic is not available, a deeper study of such traffic in comparison with those produced by voice conferencing and streaming media applications must be made.

3.3.4 Analysis of 'SMB Name Wildcard’

3.3.4.1 Alert samples

```
[**] SMB Name Wildcard [**] MY.NET.152.176:137 -> MY.NET.11.5:137
[**] SMB Name Wildcard [**] MY.NET.11.5:137 -> MY.NET.152.176:137
[**] SMB Name Wildcard [**] MY.NET.152.182:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.11.7:137 -> MY.NET.152.182:137
[**] SMB Name Wildcard [**] MY.NET.152.20:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.11.7:137 -> MY.NET.152.20:137
[**] SMB Name Wildcard [**] MY.NET.152.183:137 -> MY.NET.11.6:137
[**] SMB Name Wildcard [**] MY.NET.11.6:137 -> MY.NET.152.183:137
[**] SMB Name Wildcard [**] MY.NET.152.180:137 -> MY.NET.11.6:137
[**] SMB Name Wildcard [**] MY.NET.11.6:137 -> MY.NET.152.180:137
[**] SMB Name Wildcard [**] MY.NET.152.181:137 -> MY.NET.11.5:137

[……………………………………SNIP……………………………………]

[**] SMB Name Wildcard [**] MY.NET.152.163:137 -> MY.NET.11.5:137
[**] SMB Name Wildcard [**] MY.NET.11.5:137 -> MY.NET.152.163:137
[**] SMB Name Wildcard [**] MY.NET.152.180:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.11.7:137 -> MY.NET.152.180:137
[**] SMB Name Wildcard [**] MY.NET.152.18:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.11.7:137 -> MY.NET.152.18:137
[**] SMB Name Wildcard [**] MY.NET.152.161:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.11.7:137 -> MY.NET.152.161:137
[**] SMB Name Wildcard [**] MY.NET.152.182:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.11.7:137 -> MY.NET.152.182:137
[**] SMB Name Wildcard [**] MY.NET.152.44:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.11.7:137 -> MY.NET.152.44:137
[**] SMB Name Wildcard [**] MY.NET.152.45:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.11.7:137 -> MY.NET.152.45:137
[**] SMB Name Wildcard [**] MY.NET.152.44:137 -> MY.NET.11.5:137
[**] SMB Name Wildcard [**] MY.NET.11.5:137 -> MY.NET.152.44:137
[**] SMB Name Wildcard [**] MY.NET.152.165:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.11.7:137 -> MY.NET.152.165:137
```

Note: The timestamp has been erased from the log entries above for compact viewing.

First alert captured at: 04/12-00:00:02.664672
Last alert captured at: 04/17-23:52:40.073305

3.3.4.2 Source and destinations involved

Source Address	**Destination Address**	No. Of SMB Name Wildcard Alerts
MY.NET.152.164	MY.NET.11.6	894
MY.NET.11.6	MY.NET.152.164	887
MY.NET.11.6	MY.NET.152.157	603
MY.NET.152.157	MY.NET.11.6	598
MY.NET.11.7	MY.NET.152.183	551
MY.NET.152.183	MY.NET.11.7	550
MY.NET.152.186	MY.NET.11.6	542
MY.NET.11.6	MY.NET.152.186	540
MY.NET.11.7	MY.NET.152.19	533
MY.NET.152.19	MY.NET.11.7	523

The number of alerts being large, data has been sorted and only the top ten source and destination addresses have been displayed.

3.3.4.3 Description and analysis

This is the signature of a very sure shot attack on systems running Netbios over IP. This attack is used to enumerate Netbios information from the target host. Information such as machine name, domain name, logged on users etc. can be extracted from the target host using the SMB Name Wildcard attack. This is a valid query used by hosts to find out names and details of other hosts using Netbios over IP.

There is a certain visual basic (.vbs) script that runs on windows machines and spreads like wild fire. This script uses file shares to spread and start scanning other hosts running Netbios over IP. This script could be responsible for the alerts above. The alert could also be due to legitimate traffic.

3.3.4.4 Signs of Active Targeting

Study of the packet logs, above, indicate that the source ports and the destination ports are the same i.e. 137. If this was an instance of active targeting, then the request would, usually, come from a non -trusted port i.e. > 1024.

A specific output pattern is noticeable in the lines in the list of top ten alerts.

MY.NET.152.164	MY.NET.11.6
MY.NET.11.6	MY.NET.152.164
MY.NET.11.6	MY.NET.152.157
MY.NET.152.157	MY.NET.11.6
MY.NET.11.7	MY.NET.152.183
MY.NET.152.183	MY.NET.11.7
MY.NET.152.186	MY.NET.11.6
MY.NET.11.6	MY.NET.152.186
MY.NET.11.7	MY.NET.152.19
MY.NET.152.19	MY.NET.11.7

Notice on each alternate line the source and the destination addresses get interchanged and amongst them are hosts

MY.NET.11.6
MY.NET.11.7

Either one of these hosts are always involved in an alert. Now this is an acute problem. The script that we spoke about in the earlier section spreads through shares. The moment it is installed on a system it starts scanning the network from start to end by incrementing the host address by one and if the host address is 255 it will increment the network address by one. This goes on in an infinite loop. If this were the worm attack we would expect infected hosts in a sequence but observing the above alert table, one will notice that the infected hosts appear in a random pattern. This reduces the possibility of the alert being generated by the worm.

One needs to analyze both the addresses and understand their similarities.

```
# grep "MY.NET.11.6.*->" alert | cut -b 23-43 | sort | uniq -c
  16274  [**] SMB Name Wildca

# grep "MY.NET.11.7.*->" alert | cut -b 23-43 | sort | uniq -c
  15916  [**] SMB Name Wildca
```

Both the outputs above are of the addresses in question as sources. There is nothing in the output that we don't know about. So lets move on to check the addresses in question as destinations.

```
# grep "> MY.NET.11.6" alert | cut -b 23-43 | sort | uniq -c
  19274  [**] ICMP Echo Reque
  16206  [**] SMB Name Wildca

# grep "> MY.NET.11.7" alert | cut -b 23-43 | sort | uniq -c
  19110  [**] ICMP Echo Reque
  15914  [**] SMB Name Wildca
```

In the output above, all the alerts are unexpected. However, there is an interesting alert in which the addresses are involved. The command below is run to find more details on the ICMP Echo Request.

```
# grep "> MY.NET.11.7" alert | cut -b 23- | sort | uniq
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.10 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.11 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.12 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.13 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.14 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.157 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.158 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.159 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.15 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.160 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.161 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.162 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.163 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.164 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.165 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.166 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.167 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.168 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.169 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.16 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.170 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.171 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.172 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.173 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.174 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.175 -> MY.NET.11.7
 [**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.176 -> MY.NET.11.7
```

```
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.177 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.178 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.179 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.180 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.181 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.182 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.183 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.184 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.185 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.186 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.18 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.19 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.20 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.213 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.214 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.215 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.216 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.21 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.22 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.244 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.245 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.246 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.247 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.248 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.249 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.250 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.251 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.252 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.44 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.45 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.46 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.10 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.11 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.12 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.13 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.14 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.157 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.158 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.159 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.15 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.160 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.161 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.162 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.163 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.164 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.165 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.166 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.167 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.168 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.169 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.16 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.170 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.171 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.172 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.173 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.174 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.175 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.176 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.177 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.178 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.179 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.180 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.181 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.182 -> MY.NET.11.7
```

```
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.183 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.184 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.185 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.186 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.18 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.19 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.20 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.213 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.214 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.215 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.216 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.21 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.22 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.244 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.245 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.246 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.247 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.248 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.249 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.250 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.251 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.252 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.44 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.45 -> MY.NET.11.7
[**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.46 -> MY.NET.11.7
[**] SMB Name Wildcard [**] MY.NET.152.10:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.11:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.12:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.13:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.14:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.15:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.157:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.158:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.159:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.160:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.161:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.16:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.162:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.163:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.164:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.165:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.166:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.167:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.168:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.169:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.170:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.171:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.172:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.173:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.174:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.175:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.176:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.177:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.178:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.179:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.180:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.181:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.18:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.182:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.183:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.184:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.185:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.186:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.19:137 -> MY.NET.11.7:137
```

```
[**] SMB Name Wildcard [**] MY.NET.152.20:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.21:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.213:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.214:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.215:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.216:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.22:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.244:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.245:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.246:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.247:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.248:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.249:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.250:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.251:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.252:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.44:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.45:137 -> MY.NET.11.7:137
[**] SMB Name Wildcard [**] MY.NET.152.46:137 -> MY.NET.11.7:137
```

Note: The timestamp has been erased from the log entries above for compact viewing.

There is just a little smile on my face right now. An analysis that was going nowhere has ultimately found home. AND ITS GAME OVER!!!

My apologies for pasting a large log file but that exactly tells the tale of the alert. This sure is some kind of worm that has infected host MY.NET.11.7. Here the host scans the network using crafted packets over two rounds and then tries to make connections using 137. This is where the Snort alerts sounds 'SMB Name Wildcard'. I further went on to check the second host *.6 and found the same pattern in it.

The large file above indicates that the host; MY.NET.11.7 is infected by some worm. The host scans the network using crafted packets over two rounds and then tries to make connections using 137. This is where Snort alerts sounds 'SMB Name Wildcard. The second host *.6 was checked and the output log had the same pattern in it.

3.3.4.5 Defensive Recommendation

The exact script running on the host must be located and removed. The script behavior must be studied to see if it could infect other systems. The network must be swept and de-wormed if Script infection has spread in the network.

3.3.4.6 Correlations

There are some excellent links below on the vulnerability and the worm itself.
http://www.sans.org/newlook/resources/IDFAQ/port_137.htm

http://security.sdsc.edu/publications/network.vbs.shtml

http://www.cert.org/incident_notes/IN-2000-02.html http://lists.jammed.com/incidents/2001/05/0034.html

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

3.3.5 Analysis of 'Watchlist 000220 IL-ISDNNET-990517'

3.3.5.1 Alert Samples

```
04/12-02:46:06.918441  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.98.26:1599 -> MY.NET.150.133:1214
04/12-02:46:07.579213  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.98.26:1599 -> MY.NET.150.133:1214
04/12-02:46:08.287353  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.98.26:1599 -> MY.NET.150.133:1214
04/12-02:46:09.001148  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.98.26:1599 -> MY.NET.150.133:1214
04/12-05:31:34.870503  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.127.16:1244 -> MY.NET.150.113:1214
04/12-05:31:39.424927  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.127.16:1244 -> MY.NET.150.113:1214
04/12-05:31:44.108089  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.127.16:1244 -> MY.NET.150.113:1214
04/12-05:31:46.413362  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.127.16:1244 -> MY.NET.150.113:1214
04/12-06:49:21.985409  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.126.3:34372 -> MY.NET.150.113:1214
04/12-06:49:24.974801  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.126.3:34372 -> MY.NET.150.113:1214
04/12-06:49:25.112243  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.126.3:34372 -> MY.NET.150.113:1214
04/12-06:49:25.162134  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.126.3:34372 -> MY.NET.150.113:1214
[……………………………………SNIP……………………………………]
04/16-20:47:53.262893  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.35.8:80 -> MY.NET.153.182:1232
04/16-20:47:53.264193  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.35.8:80 -> MY.NET.153.182:1232
04/16-20:47:53.333433  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.35.8:80 -> MY.NET.153.182:1232
04/16-20:47:53.334656  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.35.8:80 -> MY.NET.153.182:1232
04/16-20:47:53.335755  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.35.8:80 -> MY.NET.153.182:1232
04/17-07:07:25.923740  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.95.23:1863 -> MY.NET.150.41:1214
04/17-07:07:28.857448  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.95.23:1863 -> MY.NET.150.41:1214
04/17-07:07:34.872245  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.95.23:1863 -> MY.NET.150.41:1214
04/17-07:07:46.891029  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.95.23:1863 -> MY.NET.150.41:1214
04/17-08:59:33.089271  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.75.145:1398 -> MY.NET.88.162:1214
04/17-08:59:41.980660  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.75.145:1398 -> MY.NET.88.162:1214
```

First alert captured at: 04/12-02:46:06.918441
Last alert captured at: 04/17-08:59:41.980660

3.3.5.2 Rule

```
alert ip 212.179.0.0/16 -> $HOME_NET any (msg:" Watchlist 000220 IL-ISDNNET-
990517")
```

3.3.5.3 Source and Destinations Involved

```
# perl alert_stats.pl "Watchlist 000220 IL-ISDNNET-990517" alert
212.179.127.31  -> MY.NET.150.41         alerts: 4
212.179.33.181  -> MY.NET.150.41         alerts: 61
212.179.33.181  -> MY.NET.88.162         alerts: 2
212.179.99.9    -> MY.NET.88.162         alerts: 2585
212.179.127.16  -> MY.NET.150.113        alerts: 4
212.179.45.202  -> MY.NET.150.113        alerts: 1
212.179.45.206  -> MY.NET.88.162         alerts: 3
212.179.125.79  -> MY.NET.150.113        alerts: 18
212.179.18.27   -> MY.NET.150.113        alerts: 3
212.179.99.10   -> MY.NET.88.162         alerts: 10600
212.179.98.200  -> MY.NET.150.220        alerts: 8
212.179.27.176  -> MY.NET.153.162        alerts: 289
212.179.27.176  -> MY.NET.153.46         alerts: 218
212.179.27.176  -> MY.NET.153.150        alerts: 194
212.179.98.224  -> MY.NET.150.220        alerts: 8
212.179.45.196  -> MY.NET.150.113        alerts: 21
212.179.27.198  -> MY.NET.150.165        alerts: 187
212.179.27.198  -> MY.NET.153.119        alerts: 21
212.179.98.4    -> MY.NET.88.162         alerts: 1
212.179.27.6    -> MY.NET.150.113        alerts: 4
212.179.27.6    -> MY.NET.88.162         alerts: 7
212.179.35.8    -> MY.NET.152.166        alerts: 18
212.179.35.8    -> MY.NET.153.210        alerts: 49
212.179.35.8    -> MY.NET.153.162        alerts: 14
212.179.35.8    -> MY.NET.153.172        alerts: 13
212.179.35.8    -> MY.NET.153.46         alerts: 13
212.179.35.8    -> MY.NET.153.182        alerts: 56
212.179.35.8    -> MY.NET.153.150        alerts: 14
212.179.127.21  -> MY.NET.150.113        alerts: 17
212.179.75.145  -> MY.NET.88.162         alerts: 2
212.179.7.38    -> MY.NET.152.215        alerts: 255
212.179.95.23   -> MY.NET.150.41         alerts: 4
212.179.7.120   -> MY.NET.150.56         alerts: 3
212.179.35.117  -> MY.NET.153.141        alerts: 6
212.179.35.118  -> MY.NET.153.162        alerts: 741
212.179.35.118  -> MY.NET.153.46         alerts: 1138
212.179.35.118  -> MY.NET.153.150        alerts: 448
212.179.35.119  -> MY.NET.153.162        alerts: 7
212.179.35.119  -> MY.NET.88.162         alerts: 2
212.179.35.119  -> MY.NET.153.150        alerts: 14
212.179.77.126  -> MY.NET.150.41         alerts: 3
212.179.54.220  -> MY.NET.150.41         alerts: 2
212.179.112.100 -> MY.NET.153.146        alerts: 291
212.179.112.100 -> MY.NET.153.196        alerts: 209
212.179.43.75   -> MY.NET.88.162         alerts: 4
```

```
212.179.43.92   -> MY.NET.150.41         alerts: 10
212.179.43.94   -> MY.NET.88.162         alerts: 3
212.179.98.26   -> MY.NET.150.133        alerts: 4
212.179.18.79   -> MY.NET.88.162         alerts: 8
212.179.44.2    -> MY.NET.150.113        alerts: 6
212.179.44.2    -> MY.NET.88.162         alerts: 8
212.179.41.241  -> MY.NET.150.113        alerts: 3
212.179.98.231  -> MY.NET.88.162         alerts: 8672
212.179.2.175   -> MY.NET.88.162         alerts: 12
212.179.97.7    -> MY.NET.88.162         alerts: 1
212.179.98.160  -> MY.NET.88.162         alerts: 44686
212.179.99.65   -> MY.NET.88.162         alerts: 4
212.179.126.3   -> MY.NET.150.113        alerts: 10
212.179.35.121  -> MY.NET.153.162        alerts: 6
212.179.35.121  -> MY.NET.153.46         alerts: 12
212.179.35.121  -> MY.NET.153.150        alerts: 6
212.179.98.254  -> MY.NET.88.162         alerts: 4510
212.179.15.28   -> MY.NET.150.220        alerts: 8
212.179.15.28   -> MY.NET.150.133        alerts: 3
```

3.3.5.4 Description and Analysis

This is a rule that generates an alert when there is any activity from 212.179.0.0/16 network. This network has grown into a hacker community since plenty of attacks have originated from that network. Lets pick up the top two address that have initiated from the network in question.

```
212.179.98.160  -> MY.NET.88.162         alerts: 44686
212.179.99.10   -> MY.NET.88.162         alerts: 10600
```

To find out more we will look at the whois report for the network.

```
# whois 212.179.98.160@whois.ripe.net
[whois.ripe.net]
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      212.179.95.0 - 212.179.99.255
netname:      CABLE-XPRMNT
descr:        Cable-Modem-Experiment
country:      IL
admin-c:      NP469-RIPE
tech-c:       NP469-RIPE
status:       ASSIGNED PA
notify:       hostmaster@isdn.net.il
mnt-by:       RIPE-NCC-NONE-MNT
changed:      hostmaster@isdn.net.il 20000103
source:       RIPE

route:        212.179.64.0/18
descr:        ISDN Net Ltd.
origin:       AS8551
notify:       hostmaster@bezeqint.net
mnt-by:       AS8551-MNT
changed:      hostmaster@bezeqint.net 20020618
```

```
source:       RIPE

person:       Nati Pinko
address:      Bezeq International
address:      40 Hashacham St.
address:      Petach Tikvah  Israel
phone:        +972 3 9257761
e-mail:       hostmaster@isdn.net.il
nic-hdl:      NP469-RIPE
changed:      registrar@ns.il 19990902
source:       RIPE
```

Whois, reports that both the addresses in question appear from the same netblock. Apparently, this block is part of a cable modem test network.

3.3.5.5 Signs of Active Targeting

Before concluding, analysis of the top two alerts must be made.

```
212.179.98.160  -> MY.NET.88.162         alerts: 44686
212.179.99.10   -> MY.NET.88.162         alerts: 10600
```

Both the alerts have a common destination address. One will have to examine the different alerts that have been registered against the host.

```
# grep "> MY.NET.88.162" alert | cut -b 23-43 | sort | uniq -c
      4  [**] EXPLOIT x86 set
      3  [**] Incomplete Pack
     25  [**] NMAP TCP ping!
     45  [**] Null scan! [**]
      2  [**] SCAN FIN [**] 1
     14  [**] SCAN FIN [**] 6
     13  [**] SCAN Synscan Po
      3  [**] SMB Name Wildca
      1  [**] SYN-FIN scan! [
     75  [**] Tiny Fragments
  71110  [**] Watchlist 00022

# grep "MY.NET.88.162.*->" alert | cut -b 23-43 | sort | uniq -c
    256  [**] ICMP Destinatio
     19  [**] ICMP Fragment R
      3  [**] SMB Name Wildca
    103  [**] spp_http_decode
```

Upon scrutiny of the above reports it is revealed that nothing very noteworthy has been established. However, a thorough study of the system, which has been accessed from the Israeli network is recommended. There is a possibility of the alerts being false positive, for legitimate connections could have given these alerts.

3.3.5.6 Defensive recommendation

The system in question should be checked for any kind of malicious activity. A thorough virus scan should be performed to sniff out Trojan code if present.

3.3.5.7 Correlations

Todd Chapman has come up with a wonderful note on the network in question in his GCIA practical available at http://www.giac.org/GCIA.php.

3.3.6 Analysis of 'spp_http_decode: IIS Unicode attack detected'

3.3.6.1 Alert samples

```
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1744 ->
211.218.200.55:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1744 ->
211.218.200.55:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1744 ->
211.218.200.55:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1744 ->
211.218.200.55:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1756 ->
218.145.54.39:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1756 ->
218.145.54.39:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1744 ->
211.218.200.55:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1744 ->
211.218.200.55:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1744 ->
211.218.200.55:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1744 ->
211.218.200.55:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1744 ->
211.218.200.55:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1744 ->
211.218.200.55:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1813 ->
211.233.29.250:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1813 ->
211.233.29.250:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.88.176:1744 ->
211.218.200.55:80

[…………………………………………SNIP…………………………………………]

[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.196:1687 ->
64.12.180.148:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.196:1688 ->
64.12.180.148:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.196:1688 ->
64.12.180.148:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.196:1688 ->
64.12.180.148:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.196:1694 ->
64.12.180.148:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.196:1694 ->
64.12.180.148:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.196:1694 ->
64.12.180.148:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.196:1695 ->
```

```
64.12.180.148:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.196:1695 ->
64.12.180.148:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.196:1695 ->
64.12.180.148:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.196:1694 ->
64.12.180.148:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.196:1694 ->
64.12.180.148:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.196:1694 ->
64.12.180.148:80
[**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.153.152:2330 ->
216.33.148.250:80
```

Note: The timestamp has been erased from the log entries above for compact viewing.

First alert captured at: 04/12-00:25:44.569832
Last alert captured at: 04/17-23:40:16.601083

3.3.6.2 Source and destinations involved

Given below are the top ten-source addresses that have generated the above alerts.

Source Address Alerts		**Destination Address** Alerts	
MY.NET.153.110	4024	211.115.213.202	3193
MY.NET.153.141	2405	211.32.117.26	2741
MY.NET.153.196	2380	211.115.218.77	2554
MY.NET.153.180	2352	211.233.29.217	1978
MY.NET.153.174	2311	211.233.29.212	1523
MY.NET.153.115	2059	211.233.28.18	1385
MY.NET.204.106	1977	211.115.213.207	1334
MY.NET.153.171	1929	211.32.117.188	936
MY.NET.153.165	1879	211.233.29.211	865
MY.NET.153.109	1817	211.233.28.104	801

Following are the top ten source and destination address that have generated the above alerts

Following are the top ten source and destination address that have generated the above alerts.

Source Address	**Destination Address**	**Alerts**
MY.NET.153.110	211.115.218.77	2554
MY.NET.153.141	211.233.29.217	1150
MY.NET.153.180	211.233.29.212	808

MY.NET.153.117	211.32.117.26	532
MY.NET.153.177	211.239.154.101	529
MY.NET.153.123	211.115.213.202	456
MY.NET.152.46	211.115.213.202	441
MY.NET.153.171	211.115.213.207	415
MY.NET.88.151	211.115.213.202	406
MY.NET.153.115	211.110.11.145	402

3.3.6.3 Description and analysis
In this attack the attacker can access any file from the server using the IUSR_machine_name account. Both IIS 4.0 and 5.0 have this vulnerability where the attacker uses '../' directory traversal exploitation if Unicode character representations are used. This vulnerability is used in the code blue worm.

Following are two examples of the exploit.

```
http://target/scripts/..%c1%1c../path/file.ext
http://target/scripts/..%c0%qf../winnt/system32/cmd.exe?/c+dir
```

Systems infected with a worm using this vulnerability scan networks to find and infect other systems running IIS 4.0 or 5.0 with the same vulnerability. The worm doesn't affect systems which do not carrying the vulnerability. The worm activity in most cases consumes a lot of precious bandwidth.

3.3.6.4 Signs of Active Targeting
This may not be a case of active targeting because this scan in most cases happens through an infected system. Attackers use tools that take advantage of this vulnerability and break integrity of a system.

3.3.6.5 Defensive recommendation
Patches and hot fixes are available from Microsoft to fix this vulnerability.

Microsoft IIS 4.0
Microsoft Patch Q269862
http://download.microsoft.com/download/winntsp/Patch/q269862/NT4ALPHA/EN-US/prmcan4i.exe
Microsoft Patch Q269862
http://download.microsoft.com/download/winntsp/Patch/q269862/NT4ALPHA/EN-US/prmcan4is.exe

Microsoft Personal Web Server 4.0

David Raitzer Patch pws_patch.zip
http://www.geocities.com/p_w_server/pws_patch/index.htm

Microsoft IIS 5.0
Microsoft Patch Q269862
http://download.microsoft.com/download/win2000platform/Patch/q269862/NT5/EN-US/Q269862_W2K_SP2_x86_en.EXE

3.3.6.6 Correlations

Heart and soul of this analysis was picked up from the bugtraq available at SecurityFocus
http://online.securityfocus.com/bid/1806

A detailed advisory is also available at http://online.securityfocus.com/advisories/2777

Microsoft's advisory on the vulnerability is also available at
http://online.securityfocus.com/advisories/2749

3.3.7 Analysis of 'SNMP public access'

3.3.7.1 Alert samples

```
[**] SNMP public access [**] MY.NET.153.178:1028 -> MY.NET.150.147:161
[**] SNMP public access [**] MY.NET.153.178:1028 -> MY.NET.150.147:161
[**] SNMP public access [**] MY.NET.153.191:1031 -> MY.NET.150.147:161
[**] SNMP public access [**] MY.NET.153.191:1031 -> MY.NET.150.147:161
[**] SNMP public access [**] MY.NET.88.203:1036 -> MY.NET.150.195:161
[**] SNMP public access [**] MY.NET.88.203:1036 -> MY.NET.150.195:161
[.............................................SNIP.............................................]
[**] SNMP public access [**] MY.NET.88.145:1044 -> MY.NET.150.195:161
[**] SNMP public access [**] MY.NET.88.145:1044 -> MY.NET.150.195:161
[**] SNMP public access [**] MY.NET.150.198:1025 -> MY.NET.113.202:161
[**] SNMP public access [**] MY.NET.150.41:1026 -> MY.NET.152.109:161
[**] SNMP public access [**] MY.NET.150.41:1026 -> MY.NET.152.109:161
[**] SNMP public access [**] MY.NET.88.207:1029 -> MY.NET.150.195:161
[**] SNMP public access [**] MY.NET.88.207:1029 -> MY.NET.150.195:161
[**] SNMP public access [**] MY.NET.150.198:1025 -> MY.NET.113.202:161
```

Note: The timestamp has been erased from the log entries above for compact viewing.

First alert captured at: 04/12-00:00:04.413810
Last alert captured at: 04/17-23:52:37.051660

3.3.7.2 Rule

```
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;
classtype:attempted-recon; sid:1411; rev:1;)

alert  tcp  $EXTERNAL_NET  any  ->  $HOME_NET  161  (msg:"SNMP  public  access  tcp";
content:"public";  reference:cve,can-2002-0012;  reference:cve,can-2002-0013;
classtype:attempted-recon; sid:1412; rev:1;)
```

The rules above generate an alert when there is an external address that sends a packet to an internal address on port 161 i.e. SNMP port that contains the public access string.

3.3.7.3 Sources and Destinations Involved

The top ten source and destination addresses that have been involved in this alert have been listed in the tables below.. Both, the source and destination addresses are internal addresses. This means that the default SNMP access rule wasn't used.

Source Address	Alerts	Destination Address	Alerts
MY.NET.70.177	14164	MY.NET.150.195	9501
MY.NET.150.198	6290	MY.NET.152.109	7325
MY.NET.150.41	2785	MY.NET.5.96	3003
MY.NET.186.10	2587	MY.NET.5.97	2976
MY.NET.153.220	1811	MY.NET.5.127	2964
MY.NET.150.245	1741	MY.NET.153.219	2625
MY.NET.88.203	1505	MY.NET.151.114	2269
MY.NET.88.159	1494	MY.NET.113.202	1798
MY.NET.88.181	1480	MY.NET.150.84	1079
MY.NET.88.136	1471	MY.NET.5.95	1050

The table below show the top ten source and destination addresses that have been involved in an SNMP public access alert.

Source Address	Destination Address	Alerts
MY.NET.70.177	MY.NET.5.96	3003
MY.NET.70.177	MY.NET.5.97	2976
MY.NET.70.177	MY.NET.5.127	2964
MY.NET.150.41	MY.NET.152.109	2785
MY.NET.186.10	MY.NET.153.219	2587
MY.NET.150.198	MY.NET.151.114	2267
MY.NET.153.220	MY.NET.152.109	1811

MY.NET.150.198	MY.NET.113.202	1798
MY.NET.150.245	MY.NET.152.109	1741
MY.NET.88.203	MY.NET.150.195	1505

3.3.7.4 Description And Analysis

SNMP is an application that provides the client with system information and statistics. In even advanced configurations SNMP can help configure a device remotely. To achieve this, SNMP uses community stings to authenticate its clients. If this string has read only access then the service will only provide information and statistics. But, if the sting has read and write access, then the string can be used to modify system parameters.

Most inexperienced administrators are least bothered about SNMP installations and securing a running SNMP service, which allows the unchanged and default community stings. This works against the system and an intruder can use default community stings for system access or even to remotely configure system parameters.

In a default installation Snort generates an alert when there is external access to port 161. But in this case, the rule seems to be modified to alert any traffic going to port 161. This is an alert that is generated when there is any host. Irrespective of internal or external, accessing the SNMP service using default community stings will sound an alert is.

3.3.7.5 Defensive recommendation

SNMP service must be dealt very carefully. This service can be a very useful service as well as can be used in a very destructive manner against the host itself.

There are newer versions of SNMP that are available where better provision has been provided for authentication and encryption.

3.3.7.6 Correlations

This paper has discussed a detect on SNMP brute force attacking. This detect can be read in section 2.5

An incident note from the GIAC that shows a sample of the SNMP public access, this note can be found at http://www.incidents.org/archives/y2k/051200.htm

Find the CVE on SNMP public access at http://icat.nist.gov/icat.cfm?cvename=CAN-2002-0012

3.3.8 Analysis of 'ICMP Echo Request L3retriever Ping'

3.3.8.1 Alert samples

```
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.176 -> MY.NET.11.5
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.182 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.20 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.183 -> MY.NET.11.6
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.180 -> MY.NET.11.6
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.181 -> MY.NET.11.5
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.158 -> MY.NET.11.6
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.215 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.177 -> MY.NET.11.6
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.21 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.46 -> MY.NET.11.6
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.164 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.174 -> MY.NET.11.6
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.22 -> MY.NET.11.7
[……………………………………SNIP……………………………………]
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.163 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.247 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.214 -> MY.NET.11.5
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.163 -> MY.NET.11.5
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.180 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.18 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.161 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.46 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.182 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.44 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.45 -> MY.NET.11.7
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.44 -> MY.NET.11.5
[**] ICMP Echo Request L3retriever Ping [**] MY.NET.152.165 -> MY.NET.11.7
```

Note: The timestamp has been erased from the log entries above for compact viewing.

First alert captured at: 04/12-00:00:02.664197
Last alert captured at: 04/17-23:52:40.072630

3.3.8.2 Rule

```
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 rule very simply puts up an alert whenever there is an ICMP packet from the external network with Type: 8 and Code: 0 (ICMP Echo Request) and contains 'ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI'

3.3.8.3 Source and destinations involved

Again the source and destination addresses are both from the internal network itself. This means either the home network isn't specified or the rule has been altered.

Source Address	Alerts	Destination Address	Alerts
MY.NET.152.164	1052	MY.NET.11.6	16297
MY.NET.152.157	900	MY.NET.11.7	15960
MY.NET.152.165	844	MY.NET.11.5	6090
MY.NET.152.12	762	MY.NET.5.4	444
MY.NET.152.183	750	MY.NET.5.92	165
MY.NET.152.177	750	MY.NET.5.96	82
MY.NET.152.168	750	MY.NET.10.49	50
MY.NET.152.179	741	MY.NET.5.142	37
MY.NET.152.13	738	MY.NET.5.3	26
MY.NET.152.181	737	MY.NET.5.35	24

The top ten source and destinations addresses involved in the L3retriver Ping alert.

Source Address	Destination Address	Alerts
MY.NET.152.164	MY.NET.11.6	889
MY.NET.152.157	MY.NET.11.6	605
MY.NET.152.186	MY.NET.11.6	547
MY.NET.152.183	MY.NET.11.7	544
MY.NET.152.19	MY.NET.11.7	528
MY.NET.152.14	MY.NET.11.7	521
MY.NET.152.46	MY.NET.11.6	520
MY.NET.152.165	MY.NET.11.7	500

MY.NET.152.168	MY.NET.11.6	493
MY.NET.152.16	MY.NET.11.7	492

3.3.8.4 Description And Analysis

L3retriver is a security analysis tool used by security analysts to audit networks. It is also a very useful penknife for attackers probing the network. The ICMP probe sends a peculiar packet containing a very unique data payload. In fact, this is one of the obvious signs of distinguishing a normal ping packet from the L3retriver ping packet.

A closer scrutiny of the top ten destinations addresses, the first two addresses have already been analyzed in section 3.3.4.4. A worm that scans networks and infects network shares infected both the hosts.

As a result, the worm throws crafted L3retriver packets into the network then its sends a crafted Hping2 or an Nmap packet and finally it will try and use Netbios to infect the system.

3.3.8.5 Defensive Recommendation

Majority of the L3retriver traffic was a result of the Netbios worm that scanned the network using L3retriver pings. There is almost no scope to find genuine L3retriver packets even if they were deliberately floated to perform recon on systems.

First of all the noise caused by the worm must be reduced by removing the worm from the network. Secondly, one must look for alerts generated by the L3retriver.

3.3.8.6 Correlations

Probably the best correlation one could find for this would be section 3.3.4 of this paper.

ArachNIDS report on the L3retriver Ping is available at
http://www.digitaltrust.it/arachnids/IDS311/event.html

The CVE for ICMP ping request from arbitrary hosts can be found at
http://icat.nist.gov/icat.cfm?cvename=CAN-1999-0523

3.3.9 Analysis of 'INFO MSN IM Chat data'

3.3.9.1 Alert samples

```
[**] INFO MSN IM Chat data [**] 64.4.12.193:1863 -> MY.NET.88.151:1445
[**] INFO MSN IM Chat data [**] MY.NET.88.151:1445 -> 64.4.12.193:1863
```

```
[**] INFO MSN IM Chat data [**] MY.NET.88.151:1445 -> 64.4.12.193:1863
[**] INFO MSN IM Chat data [**] 64.4.12.193:1863 -> MY.NET.88.151:1445
[**] INFO MSN IM Chat data [**] MY.NET.88.151:1445 -> 64.4.12.193:1863
[**] INFO MSN IM Chat data [**] 64.4.12.193:1863 -> MY.NET.88.151:1445
[**] INFO MSN IM Chat data [**] MY.NET.88.151:1445 -> 64.4.12.193:1863
[**] INFO MSN IM Chat data [**] MY.NET.88.176:1268 -> 64.4.12.178:1863
[**] INFO MSN IM Chat data [**] MY.NET.88.176:1268 -> 64.4.12.178:1863
[**] INFO MSN IM Chat data [**] MY.NET.88.176:1268 -> 64.4.12.178:1863
[**] INFO MSN IM Chat data [**] MY.NET.88.176:1268 -> 64.4.12.178:1863
[**] INFO MSN IM Chat data [**] MY.NET.88.176:1268 -> 64.4.12.178:1863
[**] INFO MSN IM Chat data [**] MY.NET.88.176:1268 -> 64.4.12.178:1863
[**] INFO MSN IM Chat data [**] MY.NET.88.176:1268 -> 64.4.12.178:1863
[**] INFO MSN IM Chat data [**] MY.NET.88.176:1268 -> 64.4.12.178:1863
[……………………………………………SNIP……………………………………………]
[**] INFO MSN IM Chat data [**] MY.NET.88.215:3532 -> 64.4.12.157:1863
[**] INFO MSN IM Chat data [**] 64.4.12.157:1863 -> MY.NET.88.215:3532
[**] INFO MSN IM Chat data [**] MY.NET.88.215:3532 -> 64.4.12.157:1863
[**] INFO MSN IM Chat data [**] MY.NET.88.215:3532 -> 64.4.12.157:1863
[**] INFO MSN IM Chat data [**] MY.NET.88.215:3532 -> 64.4.12.157:1863
[**] INFO MSN IM Chat data [**] MY.NET.88.215:3532 -> 64.4.12.157:1863
[**] INFO MSN IM Chat data [**] MY.NET.88.215:3532 -> 64.4.12.157:1863
[**] INFO MSN IM Chat data [**] 64.4.12.157:1863 -> MY.NET.88.215:3532
[**] INFO MSN IM Chat data [**] MY.NET.88.215:3532 -> 64.4.12.157:1863
[**] INFO MSN IM Chat data [**] MY.NET.151.126:3623 -> 64.4.12.181:1863
[**] INFO MSN IM Chat data [**] MY.NET.151.126:3623 -> 64.4.12.181:1863
[**] INFO MSN IM Chat data [**] MY.NET.151.126:3623 -> 64.4.12.181:1863
```

Note: The timestamp has been erased from the log entries above for compact viewing.

First alert captured at: 04/12-00:00:23.508612
Last alert captured at: 04/17-23:37:25.051269

3.3.9.2 Rule

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 1863 (msg:"INFO msn chat access"; flags:
A+; content:"text/plain"; depth:100; classtype:misc-activity; sid:540; rev:3;)
```

This rule picks up all the communication using MSN Instant Messenger i.e. any TCP packet from internal network to external network port 1863.

3.3.9.3 Source and destinations involved

Following are the top ten source and destination addresses that have been found in the alerts reported.

Source Address	Alerts	Destination Address	Alerts
MY.NET.153.211	940	MY.NET.153.111	965
MY.NET.153.111	678	MY.NET.153.211	659
64.4.12.179	632	MY.NET.153.108	657
MY.NET.153.108	588	MY.NET.153.168	601
MY.NET.153.153	569	MY.NET.151.126	449
MY.NET.153.71	485	64.4.12.179	443
64.4.12.197	475	64.4.12.197	439
MY.NET.153.168	416	64.4.12.162	431
64.4.12.159	416	64.4.12.166	420
MY.NET.153.190	413	MY.NET.153.153	355

Following are the top ten source and destination addresses that have been picked up by the rule used.

Source Address	Destination Address	Alerts
64.4.12.179	MY.NET.153.111	449
64.4.12.197	MY.NET.153.108	369
MY.NET.150.46	64.4.12.186	301
MY.NET.153.108	64.4.12.197	295
MY.NET.153.71	64.4.12.162	286
64.4.12.159	MY.NET.153.111	279
MY.NET.153.111	64.4.12.179	247

64.4.12.186	MY.NET.150.46	224
MY.NET.153.153	64.4.12.166	219
64.4.12.166	MY.NET.153.153	158

3.3.9.4 Description and analysis

This rule alerts on the use of MSN Instant Messenger. Networks in which use of instant messenger is strictly prohibited should have this rule activated.

3.3.9.5 Signs of active targeting

Signs of active targeting are extremely low because the alerts seem to be from legitimate Instant Messenger traffic. However, a Whois must be performed to find out more.

```
# whois 64.4.12.179@whois.arin.net
[whois.arin.net]
MS Hotmail (NETBLK-HOTMAIL)
   1065 La Avenida
   Mountain View, CA 94043
   US

   Netname: HOTMAIL
   Netblock: 64.4.0.0 - 64.4.63.255

   Coordinator:
      Myers, Michael  (MM520-ARIN)  icon@HOTMAIL.COM
      650-693-7072

   Domain System inverse mapping provided by:

   NS1.HOTMAIL.COM              216.200.206.140
   NS3.HOTMAIL.COM              209.185.130.68

   Record last updated on 09-Jan-2001.
   Database last updated on  9-Jul-2002 20:01:21 EDT.
```

So this traffic can be comfortably ruled out as malicious traffic.

3.3.9.6 Defensive recommendation

In case of Networks disallowing the use MSN IM, the IM software should be removed from the hosts running the client.

3.3.9.7 Correlations

David Stewart's practical has some material on the MSN IM chat alert.
http://www.giac.org/practical/david_stewart_gcia.doc

3.3.10 Analysis of 'High port 65535 udp - possible Red Worm – traffic'

3.3.10.1 Alert samples

```
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.48:55551 ->
MY.NET.153.193:65535
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.48:65535 ->
MY.NET.152.159:65280
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.50:65535 ->
MY.NET.152.175:65532
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.50:65535 ->
MY.NET.152.175:65280
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.50:65535 ->
MY.NET.152.175:65280
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.60:65535 ->
MY.NET.152.176:65408
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.50:65535 ->
MY.NET.153.195:65280
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.50:65535 ->
MY.NET.153.195:65280
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.50:65535 ->
MY.NET.153.195:65280
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.50:65535 ->
MY.NET.153.195:65408
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.53:65535 ->
MY.NET.153.163:65282
[……………………………………………SNIP……………………………………………]
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.51:65535 ->
MY.NET.152.160:65535
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.51:65535 ->
MY.NET.152.160:65535
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.52:61695 ->
MY.NET.153.211:65535
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.52:55551 ->
MY.NET.153.211:65535
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.52:55551 ->
MY.NET.153.211:65535
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.52:65535 ->
MY.NET.153.211:65535
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.52:65535 ->
MY.NET.153.211:65280
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.48:65535 ->
MY.NET.153.164:33732
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.48:65535 ->
MY.NET.153.164:33732
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.149.64:64767 -
> MY.NET.6.53:65535
[**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.6.60:59647 ->
MY.NET.153.189:65535
```

Note: The timestamp has been erased from the log entries above for compact viewing.

First alert captured at: 04/12-00:02:53.087710
Last alert captured at: 04/17-23:48:51.419840

3.3.10.2 Source and destinations involved

Following are the top ten source and destination addresses that generated the alert.

Source Address	Alerts	Destination Address	Alerts
MY.NET.6.52	3722	MY.NET.152.21	934
MY.NET.6.48	2959	MY.NET.152.157	338
MY.NET.6.50	2639	MY.NET.152.183	311
MY.NET.6.51	1496	MY.NET.152.247	292
MY.NET.6.49	676	MY.NET.152.248	287
MY.NET.6.53	339	MY.NET.152.176	249
MY.NET.6.60	295	MY.NET.153.202	237
MY.NET.6.45	158	MY.NET.152.165	220
MY.NET.60.43	133	MY.NET.153.193	195
209.249.123.160	62	MY.NET.152.184	195

These are the top ten source and destination addresses that generated the alert.

Source Address	Destination Address	Alerts
MY.NET.6.52	MY.NET.152.21	910
MY.NET.6.50	MY.NET.152.248	233
MY.NET.6.52	MY.NET.152.247	205
MY.NET.6.48	MY.NET.152.157	191
MY.NET.6.48	MY.NET.152.174	160
MY.NET.6.50	MY.NET.152.14	130
MY.NET.6.50	MY.NET.152.183	130
MY.NET.6.50	MY.NET.152.176	128

MY.NET.6.50	MY.NET.152.165	125
MY.NET.6.48	MY.NET.153.202	124

3.3.10.3 Description And Analysis

The Red Worm is the other name for Adore Worm. It scans systems for known vulnerabilities with their exploits. Exploits like BIND, Wu-FTPD, rpc-statd and LPRng are just a few. After scanning for vulnerable systems, it infects them with a torjaned version of ps. It would then send an email to three addresses that contained critical system information.

3.3.10.4 Signs of Active Targeting

Signs of active targeting are not prominent, but looking at the pattern in the table with both the source and the destination addresses, one can conclude that there is a definite worm infection.

Observe that all the source addresses are from MY.NET.6.* network. The destination addresses are also from a same network and has a somewhat sequential pattern.

3.3.10.5 Defensive Recommendation

The infected systems should be disinfected. The latest upgrades and patches should be applied to all the services and applications running on the host.

3.3.10.6 Correlations

A wonderful description of the Adore worm (Red worm) is available at the GIAC site.
http://www.sans.org/y2k/adore.htm

Redhat related information on the worm can be found at
http://www.redhat.com/support/alerts/Adore_worm.html

3.4 OOS Analysis

This analysis discusses the out of specification packets generated during the five days of log analysis. Most of the packets gathered in the OOS logs, were corrupt packets except one instance of SYN FIN Scan.

3.4.1 OOS Detect # 1

Sample log of the found packets are given below.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/13-19:22:12.075247 209.66.74.90:21 -> MY.NET.5.83:21
TCP TTL:28 TOS:0x0 ID:39426
**SF**** Seq: 0x168E05F4   Ack: 0x4CC8402C   Win: 0x404
00 00 00 00 00 00                                ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/13-19:22:12.084737 209.66.74.90:21 -> MY.NET.5.85:21
TCP TTL:28 TOS:0x0 ID:39426
**SF**** Seq: 0x168E05F4   Ack: 0x4CC8402C   Win: 0x404
00 00 00 00 00 00                                ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/13-19:22:12.123989 209.66.74.90:21 -> MY.NET.5.87:21
TCP TTL:28 TOS:0x0 ID:39426
**SF**** Seq: 0x168E05F4   Ack: 0x4CC8402C   Win: 0x404
00 00 00 00 00 00                                ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/13-19:22:12.190929 209.66.74.90:21 -> MY.NET.5.89:21
TCP TTL:28 TOS:0x0 ID:39426
**SF**** Seq: 0x168E05F4   Ack: 0x4CC8402C   Win: 0x404
00 00 00 00 00 00                                ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/13-19:22:12.191422 209.66.74.90:21 -> MY.NET.5.90:21
TCP TTL:28 TOS:0x0 ID:39426
**SF**** Seq: 0x168E05F4   Ack: 0x4CC8402C   Win: 0x404
00 00 00 00 00 00                                ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/13-19:22:12.225167 209.66.74.90:21 -> MY.NET.5.92:21
TCP TTL:28 TOS:0x0 ID:39426
**SF**** Seq: 0x168E05F4   Ack: 0x4CC8402C   Win: 0x404
00 00 00 00 00 00                                ......

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

The above packets are definitely crafted packets. All the parameters are non-variable. One needs to do a whois lookup on the source address.

```
# whois 209.66.74.90@whois.arin.net
[whois.arin.net]
Abovenet Communications, Inc. (NETBLK-NETBLK-ABOVENET2)
   Suite 1010, 50 W San Fernando,
   San Jose, CA 95113
   US

   Netname: NETBLK-ABOVENET2
   Netblock: 209.66.64.0 - 209.66.127.255
   Maintainer: ABVE

   Coordinator:
      Metromedia Fiber Networks/AboveNet  (NOC41-ORG-ARIN)  noc@ABOVE.NET
      408-367-6666
Fax- 408-367-6688

   Domain System inverse mapping provided by:

   NS.ABOVE.NET                 207.126.96.162
   NS3.ABOVE.NET                207.126.105.146

   ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
   Record last updated on 27-Apr-2001.
   Database last updated on  9-Jul-2002 20:01:21 EDT.
```

The source address seems to come from a home Internet network registered to ABOVENET2. This scan is a very peculiar scan because both, the source and destination ports are the same i.e. 21.

The source address could possibly have been spoofed because the packets are spoofed. In most cases these are packets originating from an FTP Bounce attack. The host in the internal network should be checked for scans and probes. Updated patches should be installed to stop predictive TCP sequence numbers.

3.5 Code Used For Analysis

Major part of the code used to parse the log files were from Todd Chapman's GCIA Practical. The last bit of code that converts log into a CSV was from Roland Lee's GCIA practical.

The following script was used to generate the output in the second table (Alert summary). The output was then formatted in Excel.

```
#!/usr/bin/perl -w

while(<>) {

        next unless /\[\*\*\]/;
        if (/PORTSCAN DETECTED/) {
                $alert{" PORTSCAN DETECTED"}{count}++;
                $alert{" PORTSCAN DETECTED"}{src} = 'NA';
                $alert{" PORTSCAN DETECTED"}{dst} = 'NA';
                next;
        }
        /spp_portscan/ && next;
        /\[\*\*\](.*)\[\*\*\]/ || die "Improper format!\n";
        $type = $1;
        $alert{$type}{count}++;
        /\[\*\*\].*\[\*\*\]\s+(.*\..*\..*\..*)\s->\s(.*\..*\..*\..*)/ || die
"pattern not found";
        ($src_ip, $src_port) =  split(/:/, $1);
        ($dst_ip, $dst_port) =  split(/:/, $2);
        $alert{$type}{src}{$src_ip}++;
        $alert{$type}{dst}{$dst_ip}++;
}

foreach (keys %alert) {
        $num_src = scalar(keys %{$alert{$_}{src}});
        $num_dst = scalar(keys %{$alert{$_}{dst}});
        print $alert{$_}{count}, "\t $num_src\t$num_dst\t$_\n";
}
```

The following script was used to generate the output in the third, fourth, fifth and the sixth table (Source from internal and external and destination from internal and external networks). The output was then formatted in Excel.

```
#!/usr/bin/perl -w

while(<>) {

        next unless /\[\*\*\]/;
        if (/PORTSCAN DETECTED/) {
                next;
        }
        /spp_portscan/ && next;
        /\[\*\*\].*\[\*\*\]\s+(.*\..*\..*\..*)\s->\s(.*\..*\..*\..*)/ || die
"pattern not found";
        ($src_ip, $src_port) =  split(/:/, $1);
        ($dst_ip, $dst_port) =  split(/:/, $2);
        $src{$src_ip}++;
        $dst{$dst_ip}++;
}

print "Source IP:\n\n";
foreach (keys %src) {
        print "$src{$_}\t$_\n";
}
print "\nDestination IP:\n\n";
foreach (keys %dst) {
        print "$dst{$_}\t$_\n";
}
```

The following script was used to generate the output in the top ten-destination ports table. The output was then formatted in Excel.

```
#!/usr/bin/perl -w

while(<>) {

        next unless /\[\*\*\]/;
        if (/PORTSCAN DETECTED/) {
                next;
        }
        /spp_portscan/ && next;
        /\s+(\S+)\s->\s(\S+)/ || die "pattern not found";
        #/\[\*\*\].*\[\*\*\]\s+(.*\..*\..*\..*)\s->\s(.*\..*\..*\..*)/ || die
"pattern not found";
        ($src_ip, $src_port) =  split(/:/, $1);
        ($dst_ip, $dst_port) =  split(/:/, $2);
        unless (defined($dst_port)) { $dst_port = -1; }
        $dst{$dst_port}++;
        $dst_port = undef;
}

print "\nDestination Ports:\n\n";
foreach (keys %dst) {
        print "$dst{$_}\t$_\n";
}
```

The following script was used to generate the output in the top ten port scanning sources from the internal network and from the external network table. The output was then formatted in Excel.

```
#!/usr/bin/perl -w

while(<>) {

        /\s+(\S+)\s->\s(\S+)/ || die "pattern not found";
        ($src_ip, $src_port) =  split(/:/, $1);
        ($dst_ip, $dst_port) =  split(/:/, $2);
        $scan_src{$src_ip}++;
}

foreach (keys %scan_src) {
        print "$scan_src{$_}\t$_\n";
}
```

Given below is the code alert_stat.pl

```
#!/usr/bin/perl -w

$string = shift;

while(<>) {

        next unless /$string/;
        /\s(\S+) -> (\S+)\s/ || die "pattern not found";
        ($src_ip, $src_port) =  split(/:/, $1);
        ($dst_ip, $dst_port) =  split(/:/, $2);
        $alert{$src_ip}{$dst_ip}++;
        $src_ports{$src_port}++;
        #print "$src_ip -> $dst_ip\n";
        #print "$src_ip -> $dst_ip: $alert{$src_ip}{$dst_ip}\n";
}

foreach $src (keys %alert) {
        foreach $dst (keys %{$alert{$src}}) {
                print "$src\t-> $dst\t alerts: ", $alert{$src}{$dst}, "\n";
        }
}
```

Given below is the code conver_log.pl - this script was used to parse the logs and convert them into a comma-separated file and then the .csv was imported into an Access database, which provided very valuable statistics for the practical. This code belongs to Roland Lee.

```
#
# convert_alert.pl
#
$SnortLog = 'alert';

print ("Date;Alert;Src IP;Src Port;Dst IP;Dst Port\n");
open file, "<$SnortLog";

while ($line = <file>){

   chomp $line;
   @message = split /[ ]+\[\*\*\] /, $line;
   @message1 = split /\./, @message[0];
   @message2 = split / -> /, @message[2];
   @message3 = split /:/, @message2[0];
   @message4 = split /:/, @message2[1];
   @message5 = split /-/, @message1[0];

#  This is for date format DD/MM/YYYY
   @message6 = split /\//, @message5[0];
   print ("@message6[1]/@message6[0]/2002
@message5[1];@message[1];@message3[0];@message3[1];@message4[0];@message4[1]\n");
}
close file;
```

4.0 References and Resources

Tcpdump
http://www.tcpdump.org

Snort
http://www.snort.org

MYSQL
http://www.mysql.com

Nmap
http://www.insecure.org

Incidents.Org
http://www.incidents.org

Security Focus
http://www.securityfocus.com

ISS
http://www.iss.net

CERT
http://www.cert.org

SANS
http://www.sans.org

GIAC
http://www.giac.org

Dshield
http://www.dshield.org

Eye
http://www.eeye.com

GCIA Practical by Todd Chapman
http://www.giac.org/practical/Todd_Chapman_GCIA.doc

GCIA Practical by Roland Lee
http://www.giac.org/practical/Roland_Lee_GCIA.doc

GCIA Practical by Montgomery Toren
http://www.giac.org/practical/Montgomery_Toren_GCIA.doc

Tanase, Matthew “The Future of IDS”
http://www.securityfocus.com/infocus/1518

Passive OS Fingerprinting
http://www.stearns.org/p0f/

Network Intrusion Detection – An Analyst's Handbook
Second Edition
By Stephen Northcutt and Judy Novak
Published by New Riders

Intrusion Signatures and Analysis
By Stephen Northcutt, Mark Cooper, Matt Fearnow and Karen Frederick
Published by New Riders

Snort FAQ
By Dragos Ruiu
Version 1.14 (25th March 2002)
http://www.snort.org/docs/faq.html

Snort Users Manual
By Chris Green
http://www.snort.org/docs/writing_rules/

Guide to Intrusion Detection Systems
By Brian Laing, ISS
http://www.snort.org/docs/iss-placement.pdf

Snort Database Plugin Documentation
By Roman Danyliw
http://www.snort.org/docs/snortdb/snortdb.html

SANS FAQ
Version 1.52
http://www.sans.org/newlook/resources/IDFAQ/ID_FAQ.htm