Intrusion Detection and Analysis

Sai Prasad Kesavamatham

GIAC GCIA Practical (version 3.3)
Challenge Exam
Submitted: July 7, 2003

Table of Contents

1. Assignment 1: IDS and it's Evolution

This section illustrates existing Intrusion Detection terminology and the differences between misuse detection versus anomaly detection. Issues with signature optimization and training anomaly detection engines are discussed. In the end a generic engine is suggested to interconnect various security modules.

1.1 Intrusion Detection System (IDS) – Basic Picture

Computer security is an ever-growing problem and it is here to stay. According to Computer Emergency Response Team (CERT) 42,586 incidents were reported in the 1st quarter of 2003 which is half the total number of incidents from the previous year 2002. As can be seen from the graph below the number of incidents and vulnerabilities reported has increased. This could be attributed to the growth of the internet, awareness among users, online business and easily available pre-built attack tools, as expected of a global village.

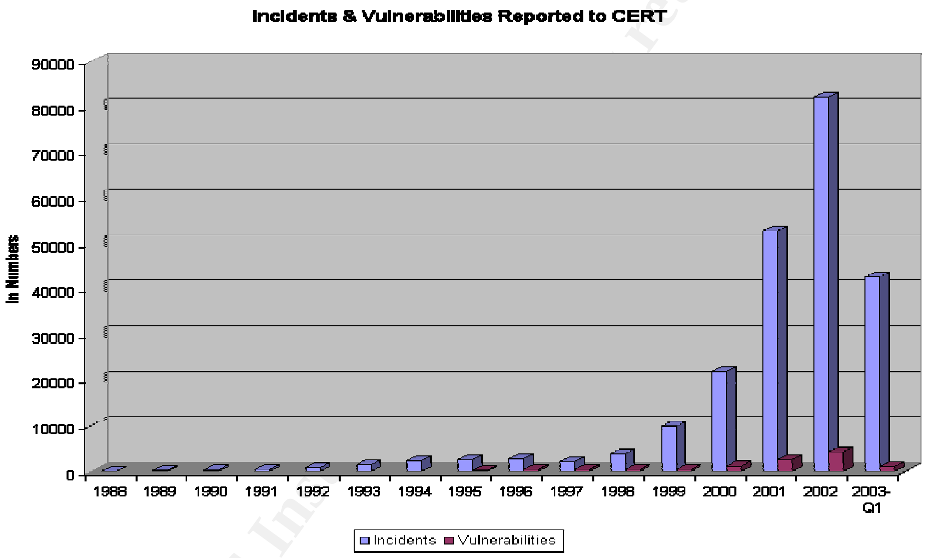


Figure 1: CERT Vulnerability Report

There are primarily three types of remotely launched attacks: probes, denial of service (DOS) and intrusions. The final goal of an intrusion detection system is to help protect the confidentiality, integrity and availability of critical information infrastructures.

The concept of 'confidentiality' can be defined as an attempt to prevent the intentional or unintentional unauthorized disclosure of information. Integrity ensures that modifications are not made by unauthorized entities and unauthorized modifications are not made by authorized entities. Availability is defined as reliable and timely access to data. Availability ensures that the systems are up and running when they are needed. The vulnerability and openness of computer systems show the necessity for monitoring both incoming and outgoing traffic. More often there is a misconception that IDS is only for incoming traffic. It is as important to monitor the outgoing traffic. After all, security is as strong as the weakest link.

Though the primary function of an IDS system has been detection, the current generation IDS systems are evolving into prevention and blocking technologies. The automated detection, immediate reporting and containment of attacks are required to respond to attacks effectively.

In a nutshell, the basic approaches to intrusion detection today may be summarized as traffic analysis using known pattern templates, state-based detection, statistical anomaly detection and other intelligent anomaly detection methods.

1.2 IDS Categorized

An Intrusion Detection System can be categorized in several ways depending on, where it is deployed in a network, how it is deployed and the approach it takes to detect intrusions. Some of the common terms used are Host based Intrusion Detection (HIDS), Network based Intrusion Detection (NIDS), Signature based IDS, Anomaly based IDS, Standalone IDS (SIDS) and Distributed IDS (DIDS).

HIDS monitors a single host and NIDS collects and analyzes all traffic on a network segment. IDS with multiple autonomous sensors or agents spread across an infrastructure can be defined as a DIDS.

1.3 IDS Approaches

1.3.1 Misuse Detection Vs Anomaly Detection

Two general approaches to intrusion detection are currently popular: Misuse detection (also loosely known as pattern matching detection) and Anomaly Detection.

Misuse detection method uses known signature pattern comparisons against the traffic. This can be done at different layers of OSI model to detect unusual traffic including layer headers and payloads. While misuse detection method can be effective in recognizing known intrusion types, it may not be satisfactory against novel attacks. There is a constant need to update and fine tune the signature database as new attacks and vulnerabilities are found. The probability for false positives is high if the signature rule sets are too generic. Signature fine tuning process is complex in a large enterprise wide network.

A false positive is defined as an alert that is raised against non-intrusive traffic. An example would be an alert raised against the viewing of a simple web page describing the signature description of a typical http attack. If the IDS signature just looks for the signature of an attack in the payload then it would raise an alert every time a user accesses this web page since the payload contained the signature.

An anomaly detection system, on the other hand, has no prior knowledge of the characteristics of possible attacks. Anomaly detection looks for patterns that deviate from normal traffic and does a statistical analysis of network traffic. But how is ‘normal traffic’ defined? One way is to train the IDS system to detect normal traffic. Anomaly detection systems have to be intelligent enough to train themselves about the normal traffic and adapt accordingly. The probability for false negatives is more under anomaly detection. However, the signature database or model can be created, managed and updated by the system intelligently.

A false negative is defined as a missed alert against a real intrusive traffic. For example, fragmenting attack data into tiny parts could fool improperly configured trivial IDS systems.

1.3.2 Sample Misuse or Pattern Matching Detection

Let’s take a look at one of the popular IDS systems, Snort. Snort versions prior to version2.0 address misuse detection or pattern matching system. Pattern matching detection is a misleading term when it comes to IDS. For example, Snort can do stateful detection across TCP streams. And it’s pattern matching capabilities go all the way into packet headers. There are around 2000 signatures or rules that come with a default installation of snort version 1.9. These rules are loaded during snort startup and arranged in a multidimensional inverted tree structure. To avoid comparing traffic against every rule, the rules are grouped into different sets and arranged.

The following simple Snort rule detects ftp root attacks. This rule tests the payload of all established TCP traffic, from any external network using any source port to the home network on destination port 21/ftp. An alert is raised if the payload content matches with the strings ‘cwd’ and ‘~root’.
RULE: alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP CWD ~root attempt"; content:"CWD "; content:" ~root"; nocase; flow:to_server ,established;)

The following example will show how a simple SYN attack can overload an IDS system. Snort 1.9 with default configuration was setup on the target machine and was assigned an IP address of 10.140.10.1. A single SYN packet was generated using the tool ‘sendip’. A spoofed IP address 10.140.10.4 was used as the source address. The attacking machine was assigned an IP address of 10.140.10.2. The SYN packet was captured on the attacking machine using tcpdump.

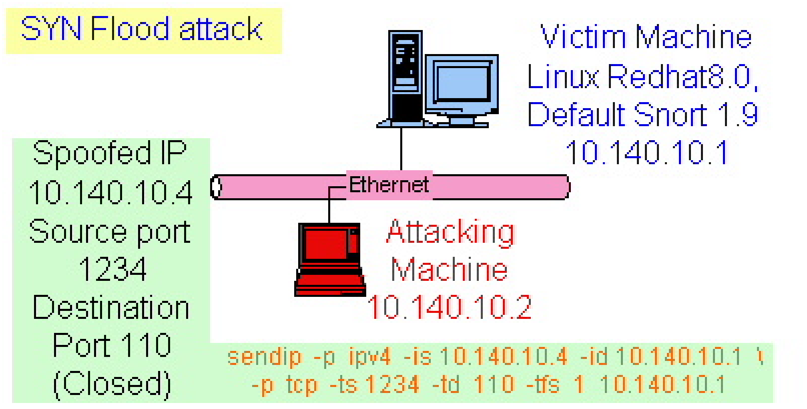


Figure 2: Lab setup for SYN flood attack

SYN packet creation using sendip:

```
# sendip –p ipv4 -is 10.140.10.4 -id 10.140.10.1 -p tcp -ts 1234 -td 110 -tfs 1
10.140.10.1
```

-p ipv4	- Use IPv4
is	- Source IP
id	- Destination IP
-p tcp	- Use tcp
-ts	- TCP Source Port
-td	- TCP Destination Port (110-pop; not open on victim's machine)
-tfs	- Set SYN flag (It is set be default, if not used)

The last field is the IP address of victim's machine.

The following tcpdump command was used to capture the SYN packet.

```
# tcpdump 'host 10.140.10.4' -w synpkt1
```

The tcpdump log can be checked using the following command.

```
# tcpdump -r synpkt1
00:01:42.306974 10.140.10.4.1234 > 10.140.10.1.pop3: S
2344219980:2344219980(0) win 65535
```

In **step1**, ran an ftp exploit in the absence of SYN attack. Established an ftp session with the victim machine and ran 'cd ~root' exploit.

```
# ftp 10.140.10.1
ftp> cd ~root
550 Unknown user name after ~
```

And the snort log file generated an alert:**[**] [1:336:5] FTP CWD ~root attempt [**]**

In **step2**, ran the same ftp exploit attack with a simultaneous SYN attack in the background. Replayed the previously captured SYN packet using tcpreplay with a loop of 500000 thus creating a SYN flood.

```
# tcpreplay -x P:1 -i eth0 synpkt1 -l 500000
sending on eth0
 500000 packets (30000000 bytes) sent in 34 seconds
 877508.1 bytes/sec 6.69 megabits/sec 14625 packets/sec
```

During the replay, ran the 'cd ~root' exploit. Snort failed to capture the alert.
*********************** **I M P O R T A N T   N O T E** ***********************
Machine configuration: 1.2GHz, 256MB and Red Hat 8.0
The above attack failed when Snort rules were optimized for ftp attack, by removing other unimportant rule sets. It has to be made clear that this example is in no way a measure of Snort's performance. This example is given to illustrate the importance of rule optimization. The burden on the hardware to maintain high-speed tcp stream state for each incoming SYN packet and to process the rules is very high.

Table 1-1: Example showing SYN flood loading

Another aspect of Snort (misuse or pattern matching detection) is the way the rules are processed. In Snort's default configuration, once a match is found against a rule, the comparison process is stopped and an alert is raised. The problem with this procedure is the probability that, a particular traffic being categorized under a 'low threat level category' thus not getting proper priority in the incident response process. An attacker may be able to fool the IDS by crafting a packet to fall in the same IDS rule chain set, triggering the low threat signatures. This results in a low priority alert, which might get overlooked in the incident response procedures. The damage could be already done by the time a second high priority alert is raised. To avoid this Snort's 'tagging' feature can be used. Snort versions 1.8 and above have a feature known as 'tagging' to avoid situations like this. Tagging allows the sensor to 'tag and follow' a host or session when a rule fires. This feature is not enabled by default.

The above discussion brings out the importance of rule set flow and optimization. The rule optimization and flow is quite different in Snort version 2.0. One of the aspects of DARPA's new 3-layered defense approach is Intrusion Tolerance. Intrusion Tolerance is the ability to continue critical operations following a partial compromise. Intrusion tolerance should be one of the key factors while designing rule optimization methods.

1.3.3 Sample Anomaly Detection

As mentioned earlier, an anomaly detection system has no prior knowledge of the characteristics of possible attacks. Anomaly detection looks for patterns that deviate from normal traffic. Anomaly detection could be used to enable machine learning to model attacks. Attack modeling could be used for creating dynamic rule sets or for observing and creating normal traffic behavior. This would further help to find out future anomalies.

Let's take a look at Spade. Spade (Statistical Packet Anomaly Detection Engine) from Silicon Defense is a snort preprocessor plug-in and can be used to monitor network traffic and classify events by assigning each event an anomaly score. Spade is one of the two components of Spice (Stealthy Portscan and Intrusion Correlation Engine).

While Snort rules look for patterns of known bad traffic, Spade raises alert when a packet crossing the network is unusual. It does this by keeping track of statistics about the traffic going past it. From this, it assigns an anomaly score to every new packet it sees. Packets with sufficiently high anomaly scores get reported. The anomaly scores are calculated using complex probability tables and binary search trees. Spade isolates all events with an anomaly score higher than a preset threshold for further correlation. Spade maintains a state file to maintain history-based probability tables and updates it periodically with current state.

In version 030125.1 of Spade, five types of detects are defined in the configuration file: "closed-dport", "dead-dest", "odd-dport", "odd-port-dest' and "odd-typecode". According to Spade usage file, these are explained as below.

> closed-dport: This is the traditional Spade detector type. This detector type is looking for packets that are going to closed ports, or at least ports that are infrequently used. This can be used to find portscans because legitimate traffic tends to go to open ports.
> dead-dest: This detector type looks for packets going to a non-alive IP address.
> odd-dport: This detector type looks for sources that are using unusual destination ports. This might indicate a compromised host or host misuse.
> odd-port-dest: This detector type looks for sources that make a connection to an unusual destination relative to what is normal for the destination port.
> odd-typecode: The detector type looks for packets with unusual ICMP type and code values.

Below is a sample alert log from Spade version 030125.1 anomaly detector.

```
[**] [104:1:1] Spade: Closed dest port used: local dest, syn: 1.0000 [**]
05/18-02:42:18.440766 68.164.67.163:2745 -> MY.HOME.NET.IP1:17300
TCP TTL:111 TOS:0x0 ID:27616 IpLen:20 DgmLen:48 DF
******S* Seq: 0xE7D31E48 Ack: 0x0 Win: 0x4000 TcpLen: 28
TCP Options (4) => MSS: 1420 NOP NOP SackOK
[**] [104:1:1] Spade: Closed dest port used: local dest, syn: 0.9045 [**]
05/18-02:42:18.983512 68.164.67.163:2745 -> MY.HOME.NET.IP1:17300
TCP TTL:111 TOS:0x0 ID:27695 IpLen:20 DgmLen:48 DF
******S* Seq: 0xE7D31E48 Ack: 0x0 Win: 0x4000 TcpLen: 28
TCP Options (4) => MSS: 1420 NOP NOP SackOK
 [**] [104:1:1] Spade: Closed dest port used: local dest, syn: 1.0000 [**]
05/18-03:50:53.931361 24.200.97.26:4165 -> MY.HOME.NET.IP2:21
TCP TTL:45 TOS:0x0 ID:1803 IpLen:20 DgmLen:52 DF
******S* Seq: 0x527D2691 Ack: 0x0 Win: 0xEBC0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP WS: 2 NOP NOP SackOK
[**] [104:1:1] Spade: Closed dest port used: local dest, syn: 0.9093 [**]
05/18-03:50:54.503409 24.200.97.26:4165 -> MY.HOME.NET.IP2:21
TCP TTL:45 TOS:0x0 ID:1926 IpLen:20 DgmLen:52 DF
```

```
******S* Seq: 0x527D2691 Ack: 0x0 Win: 0xEBC0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP WS: 2 NOP NOP SackOK
[**] [104:1:1] Spade: Closed dest port used: local dest, syn: 0.8562 [**]
05/18-03:50:54.970344 24.200.97.26:4165 -> MY.HOME.NET.IP2:21
TCP TTL:45 TOS:0x0 ID:1986 IpLen:20 DgmLen:52 DF
******S* Seq: 0x527D2691 Ack: 0x0 Win: 0xEBC0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP WS: 2 NOP NOP SackOK

[**] [104:3:1] Spade: Non-live dest used: local dest, est. flags: 1.0000 [**]
05/13-10:35:14.980352 MY.HOME.NET.IP2:14946 -> 24.60.101.200:1461
TCP TTL:128 TOS:0x0 ID:58845 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x64B832C1 Ack: 0xE10EA4D4 Win: 0xF88E TcpLen: 32
TCP Options (3) => NOP NOP TS: 4260535 15451851
```

Table 1-2: Spade Alerts

It is doubtful if Spade can be classified as a pure anomaly detection system. Spade alerts contain events and corresponding anomaly scores. The anomaly detector reports all the events with anomaly scores above the threshold level. Anomaly score is proportional to the logarithm of probability of the occurrence of the event. In table 1-2 under spade alerts the anomaly scores are highlighted. If the probability of a new event is taken as 0.1 (very low probability to happen) then the anomaly score is 1 (calculated as -log(0.1)). The anomaly score for a specific event starts at 1 and decreases with each occurrence of the event. Repeated attacks to a port bring down the anomaly score below the threshold level. It is assumed from observation that a packet to port 80 (HTTP) is more probable than to port 17300 on the same server.

Even though these detects have additional options for different flags and threshold levels, the definition and declaration of the five detect types is similar to misuse or pattern matching detection. The detects described earlier are used to define normal traffic on a network. Fundamentally, the analysis is again limited to a set of detect definitions. An ideal anomaly detection system should not have any defined rule sets. And Spade at this stage does not look to be as flexible as a signature based detection system since the above-defined rules are fixed from a user's point of view. And it is sure that there will be new detector types in future just like new signature rules.

As described earlier spade uses detector components to look for anomaly over certain types of packets. At any time more than one detector can be used. Different detectors using different detection approaches can be applied to the same set of packets. Spade also generates a probability table (history) of observed network traffic. Spade trains itself from this probability table. Below is a sample state log maintained by spade. Alerts are not generated if the history file is lost and Spade has to recreate the probability table.

```
573922 total packets were processed by spade in this run

** Closed dest port used: local dest, syn (id=1) **
```

```
55006 packets were evaluated
  144 (0.26%) packets were considered anomalous
    144 (0.26%) packets were inserted in the wait queue
    129 (0.23%) packets were reported as alerts
  177 (0.32%) packets did not have enough observations
  3240 packets were checked against the wait queue
55006 observations were stored
28861.7486 observations are remembered

** Rare dest port used: local dest, weird flags (id=2) **
0 packets were evaluated
  0 (nan%) packets were considered anomalous
    0 (nan%) packets were reported as alerts
55006 observations were stored
28861.7486 observations are remembered

** Closed dest port used: local dest, teardown flags (id=3) **
5956 packets were evaluated
  3618 (60.75%) packets were considered anomalous
    3618 (60.75%) packets were inserted in the wait queue
    775 (13.01%) packets were reported as alerts
  63 (1.06%) packets did not have enough observations
  57349 packets were checked against the wait queue
55006 observations were stored
28861.7486 observations are remembered

Other entries cut short to save space
** Closed dest port used: nonlocal dest, syn (id=4) **
2429.5499 observations are remembered
** Closed dest port used: nonlocal dest, synack (id=5) **
3090 packets were evaluated
4443 observations were stored
2429.5499 observations are remembered
** Closed dest port used: nonlocal dest, teardown flags (id=6) **
62081 packets were evaluated
4443 observations were stored
2429.5499 observations are remembered

** Non-live dest used: local dest, weird flags (id=7) **
0 packets were evaluated
  0 (nan%) packets were considered anomalous
    0 (nan%) packets were reported as alerts
85408 observations were stored
9577.2715 observations are remembered
```

Table 1-3: Spade state log

Spade can not tell if a packet is malicious much like Snort signature alert not being able to tell if an alert is a false positive. Currently Spade is good at detecting port scans and denial of service attacks.

There are other developmental/research tools like PHAD (Packet Header Anomaly Detector) to detect anomalies in Ethernet, IP, TCP, UDP, and ICMP packet headers and ALAD (Application Layer Anomaly Detector) to detect anomalies in inbound TCP stream connections to well known ports. Currently the field of anomaly detection is still very immature and has a long way to go.

1.4 Byproducts

Two other technologies that came out of IDS are Network Intrusion Prevention System (NIPS) and Honey Pots. NIDS by definition is a defensive and reactive technology. The strategy is to detect any failures in the defense system and then react to those failures. NIPS and Honey Pots go one step further.

NIPS is more than a regular NIDS. NIPS is capable of taking proactive measures instead of just being reactive to the attacks. They can coordinate with the firewalls and change the firewall rules on the fly while reacting to attacks. The problem is that they can be fooled into blocking their own systems by using simple tools like 'sendip' described above. An attacker can spoof an organization's external DNS server as the source IP address in network scans creating scan alerts or DoS alerts. Depending on the preventive rules the NIPS may respond by blocking traffic from it's own external DNS server thus denying it's own services. However this can be overcome by creating exceptions in the rule sets. But organizations have to weigh all possible scenarios before jumping on to this.

A honey pot is a system designed to be compromised. The primary purpose of a honey pot is to gather information about threats by studying new attacks as they happen. Honey pots are setup with fake data to emulate other systems, known services and vulnerabilities in a controlled environment. Known vulnerabilities are left open to attract attackers. Good data control and data capture procedures are kept in place to study the attacks. Honey net is a network of multiple honey pots and applications in a controlled environment.

1.5 Need for Standards

The Intrusion Detection field is currently moving from it's infant stage to adolescent stage. The organizational and technological instabilities are still a hurdle. One symptom of this is the lack of standards on interoperability between tools. A large company base is competing in the present IDS industry. Many well-known published pattern matching and anomaly detection methods and algorithms are being put to use to develop new IDS products.

Companies like Fidelis Security Systems provide rule optimization compilers to compile the rule base into machine code making the rule processing procedures faster. The disadvantage is recompiling the rule set for any changes in the rule set. SnortTran module works with Snort versions 1.9 to improve the performance. This is one example of a software solution.

The present leading day approach to IDS is Multi Host Network Based, also known as Distributed IDS (DIDS). As we saw above, there is a trade off for IDS speeds and accuracy. Unfortunately both are important when it comes to security. Distributed systems solve this problem. One of the significant challenges is to combine data and information from numerous heterogeneous distributed agents (and managers) into a coherent process. The study of human brain and nerve system forms a very good model to follow in this matter. Making inferences out of data from multiple and diverse sensors and sources is akin to human cognitive process where the brain fuses sensory information from various sensory organs, evaluates situations, makes decisions and directs actions.

There are, however, some attempts being made to correct this. The Intrusion Detection Working Group (IDWG), under Internet Engineering Task Force, is developing a set of functional requirements and protocols for communication between ID systems, and specifying an ID language that describes data formats. DARPA is pursuing an effort to develop a Common Intrusion Detection Framework (CIDF). This effort focuses on establishing a high-level architectural view of IDS functional components, and the data communication needs between these components. The major components are called event generators, event analyzers, event databases, and response units.

The integration of different data formats from multiple vendors, data refinement, data mining and representation is a challenge. IDWG (Intrusion Detection Working Committee) is working on the standards. IDWG described it's mission statement as below.
"The purpose of the Intrusion Detection Working Group is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems, and to management systems which may need to interact with them. The Intrusion Detection Working Group will coordinate its efforts with other IETF Working Groups."

There is a lot of potential for a field as immature, rapidly evolving, and highly competitive as the intrusion detection field.

1.6 Where is it going?

During 1995-1999 DARPA defined homeland security as a two layered approach: Strong Barriers with local detection. The new road map for 1999-2003 is a 3-layered defense approach: Intrusion Tolerance, Global Intrusion Detection and Intrusion Prevention.

Intrusion Tolerance is the ability to continue critical operations following a partial compromise.
Global Detection is the ability to distinguish events of elevated significance from those of only local interest.

A sophisticated IDS infrastructure with well-tested, established and fine tuned rule sets will still result in a large alert database thus prompting for good incidence response procedures. Too many false positives will affect the efficiency of overall escalation procedures.

More-often-than-not these systems fail to provide network engineers tangible and useful situational information, typically overwhelming operators with system messages and other low-level data. Network management and intrusion detection systems must operate in a uniform and cooperative model, fusing data into information and knowledge, so network operators can make informed decisions about the health and real-time security.

Below is a diagram showing one possible future scenario. In this, all the security devices like IDS, firewalls and honey pots are managed using one central generic engine. The generic engine has plug-ins to manage and control systems from multiple vendors and supports future IETF/IDWG standards as well for data and control management. One common management console can be used to define, create and push policies, rules and configuration changes. The real time alerts are generated by individual components and are also converted and stored in a central database. The decision-making unit can create proactive rules on the fly to change firewall rules. An integrated data mining and modeling unit analyzes the offline data and creates traffic models, anomalies and possible future proactive measures.

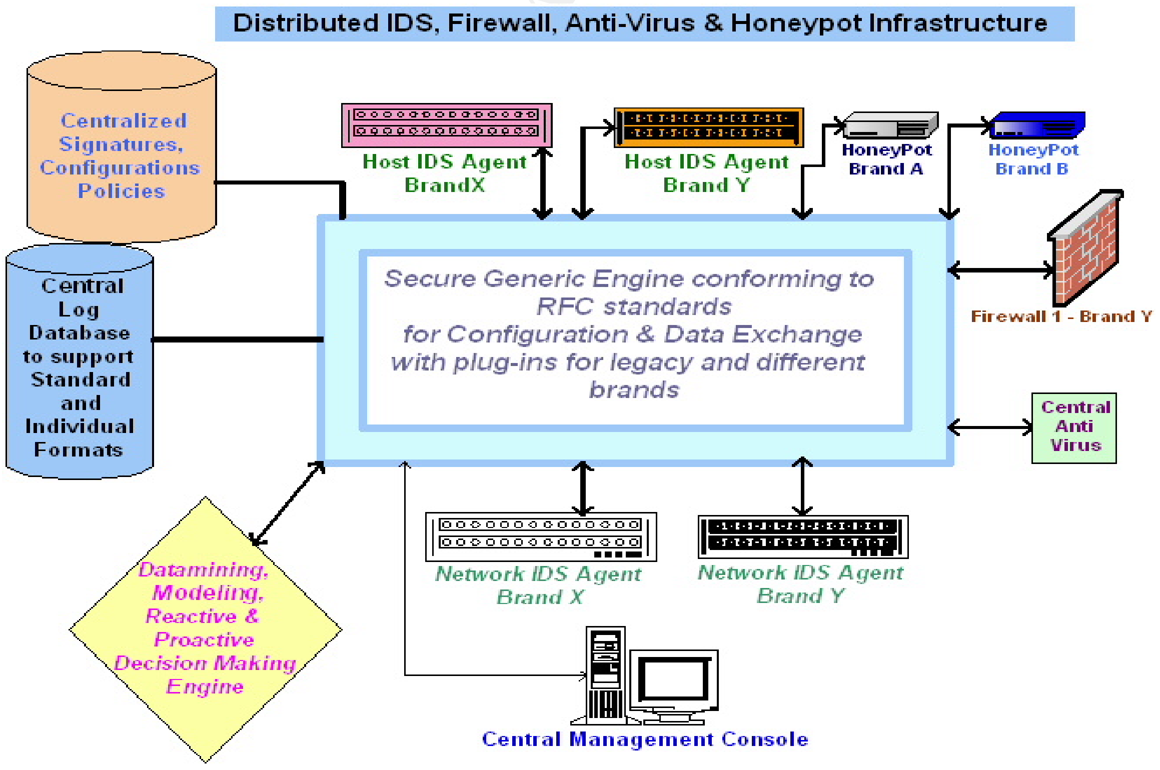

Distributed IDS, Firewall, Anti-Virus & Honeypot Infrastructure
Centralized Signatures, Configurations Policies
Central Log Database to support Standard and Individual Formats
Host IDS Agent BrandX
Host IDS Agent Brand Y
HoneyPot Brand A
HoneyPot Brand B
Firewall 1 - Brand Y
Secure Generic Engine conforming to RFC standards for Configuration & Data Exchange with plug-ins for legacy and different brands
Central Anti Virus
Datamining, Modeling, Reactive & Proactive Decision Making Engine
Network IDS Agent Brand X
Network IDS Agent Brand Y
Central Management Console

Figure 3: Future IDS

If the intrusion-detection function or the antiviral-detection function raises an attack alert, and the vulnerability-assessment function confirms that the network is vulnerable to that attack, the firewall blocks it or shunts the packet off to the honey pots, thwarting the attack.

In future, we can expect improved correlation of attacks, better proactive responses, better incident response with improved integration of system management framework, all managed from one central management console.

1.7 References:

Intrusion Detection Working Group at IETF http://www.ietf.org/html.charters/idwg-charter.html
Next Generation Cyberspace IDS
http://www.silkroad.com/paper/html/ids/node1.html
Darpa Document http://www.darpa.mil/darpatech99/presentations/itopdf/itoiis.pdf
Network Anomaly Intrusion Detection Research
http://www.cs.fit.edu/~mmahoney/dist/
Rule optimization by compiling into machine code
http://www.fidelissec.com/testdrive.html
SENDIP http://www.earth.li/projectpurple/progs/sendip.html
TCPREPLAY http://tcpreplay.sourceforge.net/
Honeynets http://www.honeynet.org
Snort http://www.snort.org
Spade http://www.silicondefense.com
http://www.sans.org/resources/idfaq/anomaly_detection.php

2. Assignment 2: Network Detects

In this section three network detects from the wild are analyzed according to the GIAC GCIA guidelines. One of the network detects was taken from http://www.incidents.org/logs/Raw. Two detects were taken from client's network. Detects were analyzed to understand the overall attack and detection mechanisms. Attack severity and correlation with other events were analyzed to evaluate risk factors and recommend counter measures for better security.

2.1 Detect1: SNMP AgenX/tcp request

2.1.1 Source of Trace

The snort logs came from a client's location. The client is running a mixed Linux/Windows environment.

2.1.2 Detect was generated by

Snort IDS, Version 1.9.0 (Build 209) with the default rule set. RedHat 7.3 with MySql Server Version 3.23.52 and ACID v0.9.6b23 were used.

The following snort rule from snmp.rules file raised this alert:

alert tcp $EXTERNAL_NET any -> $HOME_NET 705 (msg:"SNMP AgentX/tcp request"; reference:cve,CAN-2002-0012; reference:cve,CAN-2002-0013; sid:1421; rev:2; classtype:attempted-recon;)
http://www.snort.org/snort-db/sid.html?sid=1421.

This rule checks all incoming TCP traffic to destination port 705. It is interesting to note that this tcp/705 communication, as the later sections are going to reveal, is not actually between SNMP Manager and SNMP (Master) Agent but between SNMP Master Agent and other sub-agents using SNMP extension protocol, AgentX (extensible agent). This attack is basically trying to get information from Master Agent posing as a subagent. More information on AgentX is available in the next section and reference material mentioned in the reference section. The corresponding alert(s) from snorts alert log files are shown below.

```
 [**]  [1:1421:2]   SNMP AgentX/tcp request  [**]
[Classification: Attempted Information Leak]   [Priority: 2]
03/24-09:12:07.209249   203.197.254.207:49032   ->   MY.HOME.X.Y:705
TCP   TTL:38  TOS:0x0   ID:55396   IpLen:20   DgmLen:40
******S*   Seq: 0x69784748    Ack: 0x0    Win: 0x400  TcpLen: 20
[Xref => cve CAN-2002-0013][Xref => cve CAN-2002-0012]

[**]   [1:1421:2]   SNMP AgentX/tcp request  [**]
[Classification: Attempted Information Leak]   [Priority: 2]
03/24-09:12:09.433398   203.197.254.207:49033   ->  MY.HOME.X.Y:705
TCP   TTL:38   TOS:0x0   ID:55435   IpLen:20   DgmLen:40
******S*   Seq: 0x40BA2AD1    Ack: 0x0    Win: 0x400    TcpLen: 20
```

Table 2-1: Snort log entry for SNMP Agentx/tcp alert

2.1.3 Probability the source address was spoofed

This attack uses TCP and is a reconnaissance attempt. It is quite possible to send spoofed packets of this attack. But to receive any information back a sophisticated man-in-the-middle like attack needs to be performed. As can be seen from the analysis of the packets the SYN packet is set to initiate a TCP connection but there is no more traffic to further prove the completion of a TCP connection.

A dshield database check on the source address 203.197.254.207 did not ring any bells.

```
Http://www.dshield.org/warning_explanation.php?fip=203.197.254.207

Your IP (203.197.254.207) does not appear as an attacker in the DShield
database.
```

Table 2-2: Dshield database search results

The whois database shows the following data for the ownership of this IP address.

```
Inetnum: 203.197.0.0 – 203.197.255.255
netname: VSNL-IN
descr: Videsh Sanchar Nigam Ltd - India.
Descr: Videsh Sanchar Bhawan, M.G. Road
descr: Fort, Bombay 400001
country: IN
person: VSNL Tech
address: 10th Floor, 2 MG Road
address: Fort Mumbai – 400001
address: India
country: IN
```

Table 2-3: Who is database search results

The address block in Table 2-3 is assigned to Internet providers in India. Further check for this IP address in the snort logs shows that there have been similar SNMP reconnaissance attempts from this source IP. This IP address most likely belongs to a dynamic IP address range and hence may not help in finding the attacker unless coordinated with the ISP.
In the end the conclusion is that this IP address is NOT spoofed.

2.1.4 Description of the attack

SNMP (Simple Network Management) is widely used across the Internet for managing IP networks, including individual network devices. SNMP was designed in the late 80's to facilitate the exchange of management information between networked devices, operating at the application layer of the ISO/OSI model. The SNMP protocol enables network and system administrators to remotely monitor and configure devices on the network. Most current generation devices and operating systems support SNMP.

SNMP uses 161/udp for general-purpose (request/response) communications, and 162/udp for traps. Additionally, the SNMP multiplexing protocol (smux, defined in RFC1227 http://www.ietf.org/rfc/rfc1227.txt) uses 199/tcp. Another SNMP extension, the AgentX protocol (RFC2741, http://www.ietf.org/rfc/rfc2741.txt) uses 705/tcp. Request handling vulnerabilities in SNMPv1 allow remote attackers to cause a denial of service or gain privileges. AgentX attack is typically sent to TCP port 705 on victim's host.

There are plenty of different implementations. Some vendors enable SNMP by default in their devices with default access passwords and permissions. By default SNMP waits for incoming messages.

In general Agent/X attack may be preceded and/or followed by some more SNMP related attacks/reconnaissance techniques to read ('getrequest') and write ('setrequest').

Agent/X vulnerability is still considered a high level candidate at CVE (Common Vulnerabilities and Exposures). As per CVE web site, it is highly likely that this candidate will be SPLIT into multiple candidates, one or more for each vendor.

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0012

Common Vulnerabilities and Exposures (CVE) is a list or dictionary that provides common names for publicly known information security vulnerabilities and exposures. More information on this is available at http://cve.mitre.org/docs/docs2000/naming_process.html.

2.1.5 Attack Mechanism

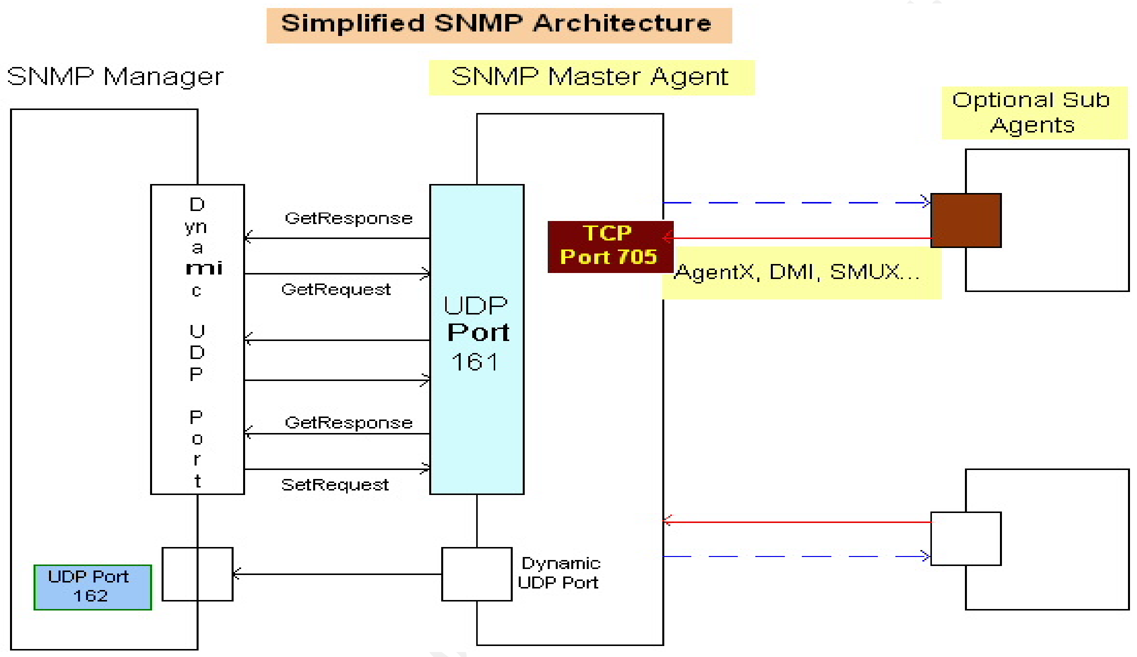


Figure 4: Simplified SNMP Architecture (From www.ee.oulu.fi Research Papers)

Before going into details of AgentX, let's look at some SNMP basics. SNMP is built around the concept of "managers" and "agents." Manager software (commonly installed on a network management system) makes requests to agent software running on a network host or device to gather data on the operational status, configuration, or performance statistics of that system (polling). Agents are configurable and allow Read access to statistics and Read/Write access to configuration parameters from management systems. Additionally, agents can send ad hoc messages as traps to manager systems informing them of unusual events.

Agents listen on 161/udp port for general-purpose (request/response) communications. Managers listen on 162/udp for traps. Additionally, one of the SNMP extensions, the AgentX protocol uses port 705/tcp.

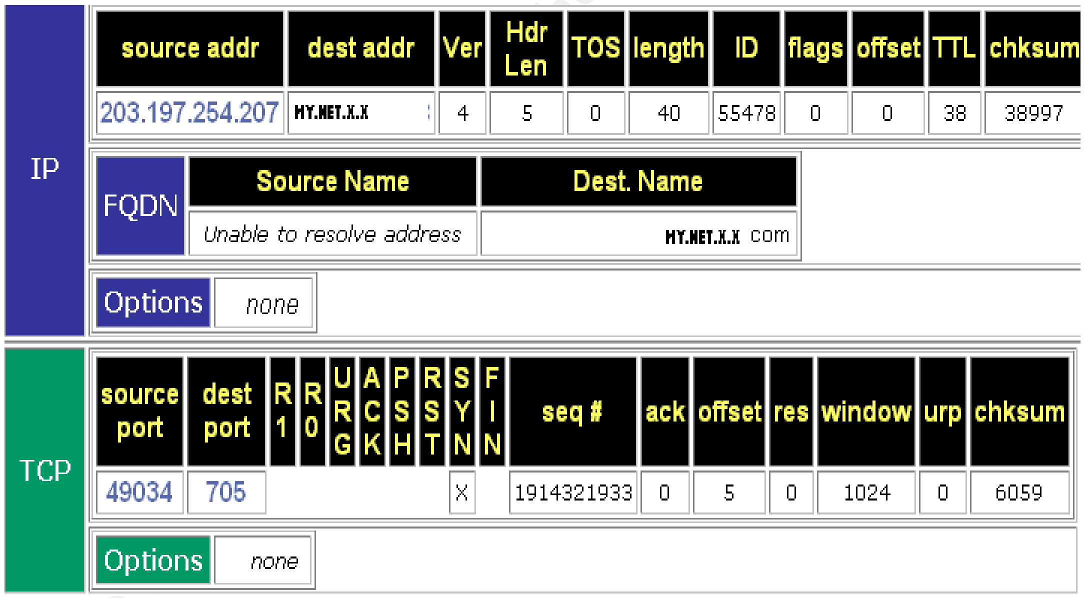
AgentX

The integration and scalability of multiple SNMP agents provided by the manufacturers and 3rd party vendors leads to a master agent and sub agents paradigm. The wide deployment of extensible SNMP agents, coupled with the lack of Internet standards in this area, makes it difficult to field SNMP manageable applications. A vendor may have to support several different subagent environments (APIs) in order to support different target platforms.

One of the protocols used between the master agent and sub agent entities is AgentX. Master agent and subagents may reside on the same host system or on a network. AgentX is a platform-independent protocol, which supports intra-agent communication within a device or local area network.

Independently developed AgentX-capable master agents and sub agents will be able to inter operate at the protocol level. Vendors can continue to differentiate their products in all other respects. Simplified, in an overall SNMP framework, a master agent appears as a single processing entity and uses SNMP protocol messages to communicate with the Manager system on one side, multiplexing the SNMP protocol messages amongst the sub agents thus shielding the subagents on the other side.

<u>AgentX over TCP</u>
The master agent accepts TCP connection requests on port 705. Subagents connect to the master agent using this port number. This following screen shot from ACID console shows the attack from source IP 203.197.254.207. The destination address has been sanitized.



IP

source addr	dest addr	Ver	Hdr Len	TOS	length	ID	flags	offset	TTL	chksum
203.197.254.207	MY.NET.X.X	4	5	0	40	55478	0	0	38	38997

FQDN	Source Name	Dest. Name
	Unable to resolve address	MY.NET.X.X com

Options	none

TCP

source port	dest port	R 1	R 0	U R G	A C K	P S H	R S T	S Y N	F I N	seq #	ack	offset	res	window	urp	chksum
49034	705							X		1914321933	0	5	0	1024	0	6059

Options	none

The destination TCP port is set to 705 and the SYN flag is set indicating the source machine's TCP connection initiation. An analysis of tcpdump log files did not reveal any response from the destination machine. This is in accordance with the site firewall rules. The firewall rules at this site are set NOT to allow any incoming packets on this port. Thus this connection attempt is a failure.

2.1.6 Correlation

As shown in Table 2-4 below, other SNMP attacks were observed from the same source address. The following excerpts are from a Snort log file.

```
[**] [1:1420:2] SNMP trap tcp [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/24-09:12:27.040139 203.197.254.207:49032 -> MY.NET.X.X:162
TCP TTL:38 TOS:0x0 ID:55835 IpLen:20 DgmLen:40
******S* Seq: 0x69784748  Ack: 0x0  Win: 0x400  TcpLen: 20
[Xref => cve CAN-2002-0013][Xref => cve CAN-2002-0012]
[**] [1:1420:2] SNMP trap tcp [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/24-09:12:29.232976 203.197.254.207:49033 -> MY.NET.X.X:162
TCP TTL:38 TOS:0x0 ID:55870 IpLen:20 DgmLen:40
******S* Seq: 0x40BA2AD1  Ack: 0x0  Win: 0x400  TcpLen: 20
[Xref => cve CAN-2002-0013][Xref => cve CAN-2002-0012]

[**] [1:1418:2] SNMP request tcp [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/24-09:15:27.113976 203.197.254.207:49032 -> MY.NET.X.X:161
TCP TTL:38 TOS:0x0 ID:60619 IpLen:20 DgmLen:40
******S* Seq: 0x69784748  Ack: 0x0  Win: 0x400  TcpLen: 20
[Xref => cve CAN-2002-0013][Xref => cve CAN-2002-0012]

[**] [1:1418:2] SNMP request tcp [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/24-09:15:28.951472 203.197.254.207:49033 -> MY.NET.X.X:161
TCP TTL:38 TOS:0x0 ID:60668 IpLen:20 DgmLen:40
******S* Seq: 0x40BA2AD1  Ack: 0x0  Win: 0x400  TcpLen: 20
[Xref => cve CAN-2002-0013][Xref => cve CAN-2002-0012]

[**] [1:1411:2] SNMP public access udp [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/24-12:41:05.457881 12.88.83.138:1368 -> MY.NET.X.X:161
UDP TTL:115 TOS:0x0 ID:38207 IpLen:20 DgmLen:72
Len: 52 [Xref => cve CAN-2002-0013][Xref => cve CAN-2002-0012]
```

Table 2-4: Snort Alerts

The above alerts were generated by the snort rules shown below.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 161 (msg:"SNMP request tcp";
reference:cve,CAN-2002-0012; reference:cve,CAN-2002-0013; sid:1418; rev:2;
classtype:attempted-recon;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 162 (msg:"SNMP trap tcp";
reference:cve,CAN-2002-0012; reference:cve,CAN-2002-0013; sid:1420; rev:2;
classtype:attempted-recon;)

snmp.rules:alert udp $EXTERNAL_NET any -> $HOME_NET 161 (msg:"SNMP
public access
```

```
 udp"; content:"public"; reference:cve,CAN-1999-0517; reference:cve,CAN-2002-
001
2; reference:cve,CAN-2002-0013; sid:1411; rev:3; classtype:attempted-recon;)
```

Table 2-5: Snort rules for alerts in Table 2-5

2.1.7 Evidence of Active Targeting

It is clear from the above logs and discussion that this is active targeting. The logs actually show many scans from this IP address on that day.
It is in fact a very long list, pages of information. Here is a glimpse.

```
Alert.0421:03/24-09:11:40.774771 203.197.254.207:49032 -> MY.NET.X.X:3128
alert.0421:[**] [117:1:1] (spp_portscan2) Portscan detected from
203.197.254.207: 1 targets 21 ports in 11 seconds [**]
alert.0421:03/24-09:11:40.777800 203.197.254.207:49032 -> MY.NET.X.X:417
alert.0421:03/24-09:11:42.977169 203.197.254.207:49033 -> MY.NET.X.X:3128
alert.0421:03/24-09:12:11.631191 203.197.254.207:49034 -> MY.NET.X.X:705
alert.0421:03/24-09:12:13.825227 203.197.254.207:49032 -> MY.NET.X.X:890
 [**] [1:1420:2] SNMP trap tcp [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/24-09:12:27.040139 203.197.254.207:49032 -> MY.NET.X.X:162
TCP TTL:38 TOS:0x0 ID:55835 IpLen:20 DgmLen:40
******S* Seq: 0x69784748  Ack: 0x0  Win: 0x400  TcpLen: 20
[**] [1:1418:2] SNMP request tcp [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/24-09:15:28.951472 203.197.254.207:49033 -> MY.NET.X.X:161
TCP TTL:38 TOS:0x0 ID:60668 IpLen:20 DgmLen:40
******S* Seq: 0x40BA2AD1  Ack: 0x0  Win: 0x400  TcpLen: 20
```

Table 2-6: Correlation – Other related alerts from 203.197.254.207

2.1.8 Severity

Severity = 0

The following formula has been used to calculate severity.
Severity = (Criticality + Lethality) –
(System Countermeasures + Network Countermeasures)

(5+4) – (4+5) = 0

Criticality	5	Criticality value is a measure of how critical the targeted system is to the network. This scan is to the gateway router, the connecting point to the internet. A compromise of this host would be disastrous.
Lethality	4	Lethality value is a measure of how severe the damage to the targeted system would be if the attack were succeeded. The result of this attack is probable revelation of sensitive information to the attacker. That would be a

		lethality value of 4 for this attack attempt.
System Countermeasure	4	This is a measure of the strength of defensive mechanism in place on the host itself. There are no logs to see the response from the system in an attack mode. But the connection was never established because of the firewall. The SNMP community passwords on the system were set using the secure password policies and as such are not easy to guess. The write access was disabled and access control list was in place to allow access only from local network. But the system was not specifically tested for SNMP attacks. Being paranoia would give this a value of 4
Network Countermeasures	5	This is a measure of the strength of defensive mechanism in place on the network. The incoming UDP/TCP traffic for all SNMP related traffic is blocked/dropped at the firewall.

Table 2-7: Severity Calculation Table

* Each value is ranked on a scale of 1-5, 1being the least and 5 the highest.

2.1.9 Defensive Recommendation

As a defensive measure

- Disable SNMP if not absolutely required.
- Update SNMP to latest available version and apply vendor patches
- Use the same policy for community names as used for passwords.
- Reassess SNMP community password string and access control lists.
- Block the traffic to the following ports at network ingress point. Basic router Access Control Lists (ACL) can help put these simple filters in place.

snmp	161/udp	# Simple Network Management Protocol (SNMP)
snmp	161/tcp	# Simple Network Management Protocol (SNMP)
snmp	162/udp	# SNMP system management messages
snmp	162/tcp	# SNMP system management messages
smux	199/tcp	# SNMP Unix Multiplexer
smux	199/udp	# SNMP Unix Multiplexer
synoptics-relay	391/tcp	# SynOptics SNMP Relay Port
synoptics-relay	391/udp	# SynOptics SNMP Relay Port
agentx	705/tcp	# AgentX
snmp-tcp-port	1993/tcp	# cisco SNMP TCP port
snmp-tcp-port	1993/udp	# cisco SNMP TCP port

- Make MIBs (Management Information Base) read only where possible. The information that SNMP agent forwards to network management console is contained in a Management Information Base (MIB). A MIB is a configuration that defines what information can be obtained from a network device, and what functions can be controlled.

More information on MIB is available at https://www.juniper.net/techpubs/software/junos/junos55/swconfig55-net-mgmt/html/snmp-overview3.html#1025454

- If possible test the SNMP infrastructure using the PROTOS SNMP test suite.
- AgentX Security Considerations:

Enable Agent/x over UNIX-domain Sockets if possible. An AgentX subagent can connect to a master agent using either a network transport mechanism (e.g., TCP), or a "local" mechanism (e.g., shared memory, named pipes). In the case where a local transport mechanism is used and both subagent and master agent are running on the same host, connection authorization can be delegated to the operating system features. The operating system will allow the connection only if the subagent has sufficient privileges.

Currently there is no access control mechanism defined in AgentX. Also there is no inherent security if a network transport is used. Transport Layer Security, SSL, or IPsec SHOULD be used to control and protect subagent connections in this mode of operation. RFC 2741 clearly recommends that subagents always run on the same host as the master agent and that operating system features be used to ensure that only properly authorized subagents can establish connections to the master agent.

2.1.10 Multiple Choice Test Question

Q) Which NET-SNMP tool kit command can be used to retrieve lot of information at once?

A) snmpget
B) snmptrap
C) snmpwalk
D) snmptranslate

Answer) **C**

Q) Which of the following NET-SNMP toolkit command(s) show(s) all the packages installed on a Red Hat 8.0 machine (enabled with SNMP)?

A) snmpwalk -v 1 -c <CommunityString> <HostName>
B) snmpwalk -v 1 -c <CommunityString> <HostName> hrSWInstalledName
C) snmpwalk -v 1 -c <CommunityString> <HostName> HOST-RESOURCES-MIB::hrSWInstalledName
D) snmpget -v 1 -c <CommunityString> <HostName> hrSWInstalledName

Answer) **B & C**

2.1.11 References:

http://www.cert.org/advisories/CA-2002-03.html

http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/snmp.htm#xtocid210315
http://www.ietf.org/rfc/rfc2741.txt?number=2741
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0012
http://www.cert.org/tech_tips/snmp_faq.html
http://www.sans.org/top20/top10.php
https://www.juniper.net/techpubs/software/junos/junos55/swconfig55-net-mgmt/html/snmp-overview3.html#1025454
http://www.ee.oulu.fi/research/ouspg/protos/testing/c06/snmpv1/
http://www.nwfusion.com/archive/1999/77058_10-04-1999.html
http://www.net-snmp.org/

2.2 Detect2: MS-SQL Worm Propagation Attempt

2.2.1 Source of Trace

The snort logs came from a small business client's location. The client is running Linux and Windows based mixed environment.

2.2.2 Detect was generated by

Snort IDS, Version 2.0.0 (Build 72) with the default rule set. Used RedHat 7.3 with MySql Server Ver 11.15 Distrib 3.23.41, for redhat-linux-gnu (i386) and ACID v0.9.6b21. The snort rule raising this alert is from sql.rules file:

sql.rules:alert udp $EXTERNAL_NET any -> $HOME_NET 1434 (msg:"MS-SQL Worm propagation attempt"; content:"|04|"; depth:1; content:"|81 F1 03 01 04 9B 81 F1 01|"; content:"sock"; content:"send"; reference:bugtraq,5310; classtype:misc-attack; reference:bugtraq,5311; reference:url,vil.nai.com/vil/content/v_99992.htm; sid:2 003; rev:2;)
http://www.snort.org/snort-db/sid.html?sid=2003

The above rule checks all incoming UDP traffic to destination port 1434. The keyword 'content' allows the user to set rules that search for specific content in the packet payload and trigger response based on that data. The option data for the content keyword can contain mixed text and binary data. The binary data is generally enclosed within the pipe (|) character and represented as bytecode. Bytecode represents binary data as hexadecimal numbers and is a good shorthand method for describing complex binary data.
Depth is a content rule option modifier. This sets the maximum search depth for the content pattern match. In the above rule the first byte of the payload is tested for '04'. Then the payload is also tested for the content 'sock', 'send' and binary '81 F1 03 01 04 9b 81 F1 01'. This last binary content is equal to 'sendso ck'.
More information on Snort rules can be found at
http://www.snort.org/docs/writing_rules/

The corresponding alert(s) from Snort alert logs are shown below.

```
[**] [1:2003:2] MS-SQL Worm propagation attempt [**]
```

```
[Classification: Misc. Attack] [Priority: 2]
05/06-08:11:58.717043 216.32.144.19:1404 -> xxx.xxx.xxx.19:1434
UDP TTL:107 TOS:0x0 ID:65475 IpLen:20 DgmLen:404 Len: 376
[Xref => http://vil.nai.com/vil/content/v_99992.htm][Xref => http://www.security
focus.com/bid/5311][Xref => http://www.securityfocus.com/bid/5310]
```

Table 2-8: Snort Alerts

2.2.3 Probability the source address was spoofed

This attack uses UDP and as such it is easy to spoof attacks using UDP since there is no need for establishing a connection before starting to exchange information. However as described in later sections, this attack normally comes from an affected machine and the victim IP addresses are randomly generated. So there is a very high probability that this IP address is not spoofed. But it is possible to generate this type of attacks with spoofed addresses using tools like sendip.

Using the following sendip command, the attack can be simulated very easily evoking an alert response from Snort.

```
# sendip -p ipv4 -is 24.60.181.163 -id 12.235.69.190 -p udp -us 2001 -ud 1434
12.235.69.190 -d 0x0481F10301049B81F10173656e64736f636b
81 F1 03 01 9B 81 F1 01 is equal to s e n d s o c k
```

A further analysis of tcpdump logs does not reveal any information not found in the Snort log. No additional packets were exchanged during this time as seen from the tcpdump logs. And this conforms to packet filtering configurations setup on the Internet connection of the network. The filtering rules on the firewall are set to block tcp connections to this port from outside.

```
Using tcpdump
# /usr/sbin/tcpdump -nn -r dumplog101 host '216.32.144.19'
0x0000   4500 0194 d4f2 0000 7011 b989 d820 9013       E.......p.......
0x0010   0ceb 45be 0803 059a 0180 726b 0401 0101       ..E.......rk....
0x0020   0101 0101 0101 0101 0101 0101 0101 0101       ................
0x0030   0101 0101 0101 0101 0101 0101 0101 0101       ................
0x0040   0101 0101 0101 0101 0101 0101 0101 0101       ................
0x0050   0101                                          ..
This does not show anything additional except the initial few bytes of the attack.
The attack starts with a 04 followed by a string 01s. More on it later.
```

Table 2-9: Tcpdump

A quick check of the source address 216.32.144.19 in the dshield database came with a positive result.

```
http://www.dshield.org/warning_explanation.php?fip=216.32.144.19
Your IP (216.32.144.19) appears as an attacker 25,246 times in the DShield
database.
```

Table 2-10: Dshield Results

The whois database (http://www.whois.sc) showed the following data for the ownership of this IP address. This is most probably an infected machine.

```
NetRange: 216.32.0.0 - 216.35.255.255          CIDR: 216.32.0.0/14
OrgName: Cable & Wireless
OrgID: EXCW
Address: 3300 Regency Pkwy City: Cary StateProv: NC PostalCode: 27511
Country: US
NetType: Direct Allocation
```

Table 2-11: Who is database search

As shown at http://www.boredom.org/~cstone/worm-annotated.txt an ‘objdump’ code analysis of this traffic reveals that the code never generates a different source IP address

```
# objdump -s -m i386 -b binary -z --disassemble-all --start-addres 0xc0 \
--show-raw-insn onepacket.txt
```

The tool ‘objdump’ shows the code for the packet at assembly language level. From the above analysis it is concluded that this is not a spoofed address. More information on ‘objdump’ is available at http://www.boredom.org/~cstone/worm-annotated.txt.

2.2.4 Description of the attack

The malicious code exploits vulnerability in Microsoft SQL Server. The code instructs the server to go into an endless loop, continually sending out data to other computers.

This worm tries to exploit vulnerability in Microsoft SQL-Server's 'Monitor Port', making it possible for remote users to execute arbitrary code. The monitor port is used to discover the connection methods offered by a particular server. The message sent by the client is usually a single byte (0x02). The response depends on the server's configuration. David Lichtfield (http://www.ngssoftware.com/advisories/mssql-udp.txt) discovered two buffer overflow conditions in this service, which can be used to execute arbitrary code in the security context of the server.

Remote users could gain access by exploiting the ‘heap’ based overflow in SQL Server Resolution Service. This virus exists only in memory of unpatched Microsoft SQL servers. Its purpose is simply to spread from one system to another and it does not carry a destructive payload.

This worm causes increased traffic on UDP port 1434 and spreads between SQL servers. A network sniffer can easily monitor this traffic. Heavy network traffic, associated with this threat, can affect network performance on all systems on the network.

http://www.securityfocus.com/bid/5310/discussion/
http://vil.nai.com/vil/content/v_99992.htm

2.2.5 Attack Mechanism

Microsoft SQL Server uses SQL Monitor service or resolution service running on UDP port 1434 to guide incoming client connections to connect to the SQL server running on TCP port 1433. The clients can typically connect using either named pipes over NetBIOS session (TCP port 139/445) or sockets with clients connecting to TCP port 1433 or both. Typically clients send messages to a UDP port 1434 to dynamically discover how the client should connect to the Server. Below is a single byte packet.
0x02 is used to discover the connection type.
0x04 is used to request to open a registry entry.
0x0A is used to ping the service and responded by a 0x0A to the source and port address.

When the first a 0x04 is sent in the payload as the first byte, the SQL Monitor thread takes the remaining data in the packet and attempts to open a registry key using this user supplied information. For example, by sending \x04\x41\x41\x41\x41 (0x04 followed by 4 upper case 'A's) SQL Server attempts to open
'HKLM\Software\Microsoft\Microsoft SQL Server\AAAA\MSSQLServer\CurrentVersion'.

In the attack described, the worm body starts with byte 0x04 (followed by a long series of 01s) which when received by the SQL Monitor generates a long registry key name (HKLM\Software\Microsoft\Microsoft SQL Server\.....\MSSQLServer\CurrentVersion) overflowing the buffer (dots in this key denote the Registry key branch SQL Monitor tries to access - for SQLSlammer.worm these bytes are are a long series of unprintable 01 symbols following 04 which is a type of request described earlier). That overwrites the return address on stack and the worm code receives control with the privileges of the SQL Monitor. The vulnerability that this worm uses lays in the SSNETLIB.DLL in the routine that handles requests of type 04 to 1434/udp port.

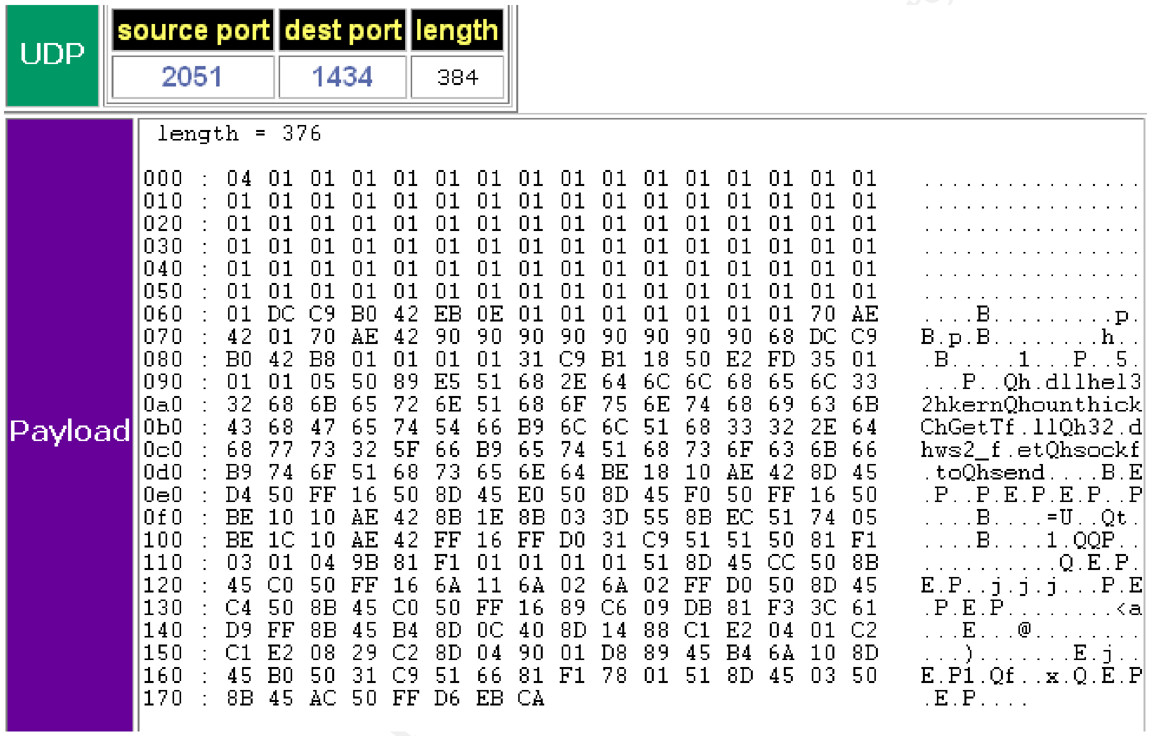
Once the worm gets control on the target computer it loads WS2_32.DLL and starts to continually send itself to port 1434/udp of randomly selected IP targets in an infinite loop. The IP of a victim is constructed using 'GetTickCount' API and is purely random. There is no skew towards the local subnet, for example. This propagation strategy consumes a lot of network bandwidth because the vast majority of requests go to the Internet.

The worm does no error checking whatsoever - its operation may cause crashes of SQL servers it was not designed for. The worm exists only in memory and does not modify any local files.

This worm would cause increased traffic on UDP port 1434. This could result in heavy network traffic and can effect network performance of systems on the network.

As per Network Associates, Inc., the malformed packet is only 376 bytes long and carries the following strings: "h.dllhel32hkernQhounthickChGetTf", "hws2", "Qhsockf" and "toQhsend" in the payload.

The following is a screen shot of payload from ACID console.



UDP	source port	dest port	length
	2051	1434	384

Payload

```
 length = 376

000 : 04 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
010 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
020 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
030 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
040 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
050 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
060 : 01 DC C9 B0 42 EB 0E 01 01 01 01 01 01 01 70 AE   ....B.........p.
070 : 42 01 70 AE 42 90 90 90 90 90 90 90 90 68 DC C9   B.p.B........h..
080 : B0 42 B8 01 01 01 01 31 C9 B1 18 50 E2 FD 35 01   .B.....1...P..5.
090 : 01 01 05 50 89 E5 51 68 2E 64 6C 6C 68 65 6C 33   ...P..Qh.dllhel3
0a0 : 32 68 6B 65 72 6E 51 68 6F 75 6E 74 68 69 63 6B   2hkernQhounthick
0b0 : 43 68 47 65 74 54 66 B9 6C 6C 51 68 33 32 2E 64   ChGetTf.llQh32.d
0c0 : 68 77 73 32 5F 66 B9 65 74 51 68 73 6F 63 6B 66   hws2_f.etQhsockf
0d0 : B9 74 6F 51 68 73 65 6E 64 BE 18 10 AE 42 8D 45   .toQhsend....B.E
0e0 : D4 50 FF 16 50 8D 45 E0 50 8D 45 F0 50 FF 16 50   .P..P.E.P.E.P..P
0f0 : BE 10 10 AE 42 8B 1E 8B 03 3D 55 8B EC 51 74 05   ....B....=U..Qt.
100 : BE 1C 10 AE 42 FF 16 FF D0 31 C9 51 51 50 81 F1   ....B....1.QQP..
110 : 03 01 04 9B 81 F1 01 01 01 01 51 8D 45 CC 50 8B   ..........Q.E.P.
120 : 45 C0 50 FF 16 6A 11 6A 02 6A 02 FF D0 50 8D 45   E.P..j.j.j...P.E
130 : C4 50 8B 45 C0 50 FF 16 89 C6 09 DB 81 F3 3C 61   .P.E.P........<a
140 : D9 FF 8B 45 B4 8D 0C 40 8D 14 88 C1 E2 04 01 C2   ...E...@........
150 : C1 E2 08 29 C2 8D 04 90 01 D8 89 45 B4 6A 10 8D   ...).......E.j..
160 : 45 B0 50 31 C9 51 66 81 F1 78 01 51 8D 45 03 50   E.P1.Qf..x.Q.E.P
170 : 8B 45 AC 50 FF D6 EB CA                           .E.P....
```

As can be seen, the protocol used is UDP and the destination port is 1434, default port for Microsoft SQL Server. The first byte is 04, followed by a long series of 01s. The strings “h.dllhel32hkernQhounthickChGetTf", "hws2", "Qhsockf" and "toQhsend” can also be seen in the payload.

2.2.6 Correlation

Though this exploit hit the Internet very recently (January 25th, 2003), it has it’s roots in the past associated with other Microsoft SQL server vulnerabilities. More information is available at http://www.securityfocus.com/bid/5310/credit/. MS-SQL related attacks are still on the top list at http://isc.incidents.org/.

No correlation was found with other GCIA papers possibly because of this vulnerability’s recent birth and since the recent papers were not yet posted to the GIAC web site at the time of writing this paper.

Some related discussion group messages on January 25th, 2003 are available at http://www.broadbandreports.com/forum/remark,5771929~root=security,1~mode=flat

2.2.7 Evidence of Active Targeting

As per attack descriptions observed so far the destination IP addresses are generated at random. The following IP addresses from the alert logs matched with dshield (http://www.dshield.org) database for MS SQL Worm. It is concluded that there is no active targeting involved.

66.50.6.99, 67.39.162.209, 80.110.49.49, 130.94.135.7, 211.93.19.50, 211.147.61.196, 216.32.144.19, 202.108.158.99. These machines are most probably compromised machines and the attacks are probably random. No scans were found from these addresses during this time.

'Who is' information for some of the address from the above list as found at http://www.whois.sc is given below.

66.50.6.99 Puerto Rico Telephone Company PRTC-NET (NET-66-50-0-0-1) 66.50.0.0 - 66.50.255.255 PRTC RAS PRTC-66-50-0-0 (NET-66-50-0-0-2) 66.50.0.0 - 66.50.30.255 **216.32.144.19** OrgName: Cable & Wireless OrgID: EXCW Address: 3300 Regency Pkwy City: Cary StateProv: NC PostalCode: 27511 Country: US CIDR: 216.32.0.0/14 **211.93.19.50** inetnum: 211.93.16.0 - 211.93.19.255 netname: XJWL descr: wl city ,Xin Jiang Uigur autonomous region POP, China.P.R country: CN

Table 2-12: Who is database search results

2.2.8 Severity

Severity = -3

The following formula has been used to calculate severity.

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)
(0+3) – (5+3)

Criticality	0	Criticality value is a measure of how critical the targeted system is to the network. This attack is sent to the border router connecting the internet and did not get forwarded to the internal network. As such it is not affected by this attack.
Lethality	3	This warm caused havoc in the first few hours on the internet but it's effect went down because of the immediate collective response from the internet. There was no UPD storm on client's network.

System Countermeasures	5	The system was patched with SP3, so there was no direct threat to the system in this scenario.
Network Countermeasures	3	The incoming UDP traffic is blocked at the firewall. But the packet filtering did overlook the traffic from DNS servers. That leaves the system open to an attacker trying to target this particular system.

Table 2-13: Severity Calculation Table

* Each value is ranked on a scale of 1-5, 1being the least and 5 the highest.

2.2.9 Defensive Recommendation

Exploitation of these security holes goes over UDP, a connection-less communications protocol. As such it makes the task of bypassing the protection offered by a firewall considerably easier. The spoofing of an IP address in a UDP packet is also considerably easier.

It is trivial for an attacker to send an attack through the firewall, setting the source IP address to that of site's external DNS Server and the source port to 53. Most firewalls will pass this packet as this packet appears like a response to a query to resolve a domain name.

- Setup a firewall rule to drop any incoming packets to port 1433, 1434, 1435 over UDP. No hosts including DNS servers should be allowed to send traffic to this port.
- Setup a firewall rule to drop spoofed packets with source address set to an internal address.
- Apply the latest patches for the Operating system and MS SQL Server.
- NGSSoftware recommends running the SQL Server as low privileged local account and not SYSTEM or a domain account.
- Since this worm lives only in memory, reboot the machine before applying the patch.
- Use a network traffic analyzer to identify the internal infected hosts.

2.2.10 Multiple Choice Test Question

Q) Choose the best way to prevent SYN attacks against public servers directly facing the Internet.

A) Increase the size of the connection queue (SYN ACK queue).
B) Decrease the time-out waiting for the three-way handshake.
C) Employ vendor software patches to detect and circumvent the problem (if available).
D) All of the above

Answer: **D**

2.2.11 References:

http://www.boredom.org/~cstone/worm-annotated.txt
http://www.ngssoftware.com/advisories/mssql-udp.txt
http://www.securityfocus.com/bid/5310/discussion/
http://vil.nai.com/vil/content/v_99992.htm
http://securityresponse.symantec.com/avcenter/Analysis-SQLExp.pdf
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-039.asp
http://www.cert.org/tech_tips/snmp_faq.html

2.3 Detect3: BACKDOOR Q access

2.3.1 Source of Trace

The data file used was obtained from http://www.incidents.org/logs/Raw/. The file 2002.10.1 was used for this analysis. Network 207.166.0.0 has been determined as the home network range. The presence of the source MAC address 00:03:e3:d9:26:c0 and destination MAC address 00:00:0c:04:64:33 on all packets belong to ranges assigned to Cisco Systems. It is assumed that the sensor is placed between two Cisco routers facing the internet and internal network.

This detect was posted to intrusions@incidents.org message group as per the practical guidelines. It is also available at this link
http://cert.uni-stuttgart.de/archive/intrusions/2003/06/msg00166.html
All the messages posted by group members in regards to this detect are shown at the end of this section. Due credit is given to all those members who took their time to read and respond to my posting.

2.3.2 Detect was generated by

This detect was generated by Snort IDS, Version 2.0.0 (Build 72) with default rule sets. Used RedHat 8.0 with MYSQL Server Version 3.23.52 and ACID v0.9.6b23.

The file was processed using the command line:

```
# snort  -c  /tmp/detects/snort.conf  -k  none  -r  /tmp/detects/2002.10.1
```

A brief explanation of the options follows:
-c specify location of the Snort configuration file
-k sets checksum checking off altogether since checksums have been altered in the logs
-r read data from the file specified

The snort rule raising this alert is from ‘backdoor.rules’ file:

alert tcp 255.255.255.0/24 any -> $HOME_NET any (msg:"BACKDOOR Q access"; flags:A+; dsize: >1; reference:arachnids,203; sid:184; classtype:misc-activity; rev:3;)
Snort Rule Ref: http://www.snort.org/snort-db/sid.html?sid=184

The Snort signature shown above raises an alert for any TCP traffic from 255.255.255.0/24 originating from any source port to the home network on any destination port. The TCP flags are tested to see if the TCP-ACK (acknowledgement) flag is set with optional other flag combinations set. The data payload size has to be of at least 1 byte.

A sample alert from the log file:

```
[**] [1:184:3] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
11/09-21:59:13.066507 255.255.255.255:31337 -> 207.166.143.84:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
```

2.3.3 Probability the source address was spoofed

This source address 255.255.255.255 was spoofed. By default, most of the IP routers do not forward network broadcast packets. A good TCP/IP stack should not reply to this address in the first place. And the presence of ACK and RST flags in the packet indicate a non-graceful abort of the session, either to indicate non-existence of any listening process or to indicate that the arriving packet does not belong to an established connection.
When the target (victim) machine replies to this source address it is replying to a broadcast address thus creating a packet storm, kind of denial of service. Of course, in this case the storm is limited to the local network. Since the source address in this alert is a broadcast address it is very easy to spoof the source address. And there is absolutely no need to use the real IP address while using or looking for Q.

(*Note: This detect was posted on 6/13/2003, 10:55 PM to the intrusions discussion forum at intrusions@incidents.org as per the assignment requirements. Excerpts from the discussion related to the post are attached at the end of this section.)

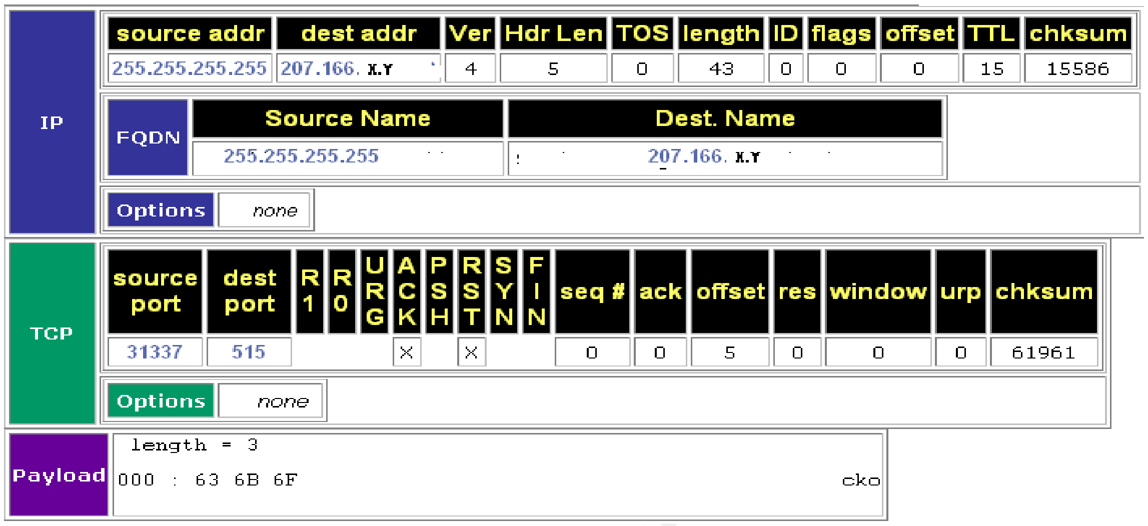
As seen in the description of the attack section below, a backdoor Q client does not necessarily need a reply from the victim machine (Q server). Q is capable of relaying or bouncing to a different IP address or network and it is very easy to send a control command from the client and to direct the victim machine to connect to an entirely different machine.

As per the README file at http://www.incidents.org/logs/Raw/README, 'the log files are the result of a Snort instance running in binary logging mode. This means that only the packets that violate the ruleset appear in the log. The logs themselves have been sanitized and the checksums have been modified'. That makes things little tricky.

2.3.4 Description of the attack

The attack appears like commands are being sent to a compromised host on the Home Network. The analysis of the attack shows 'Q' like activity with a possible Q server running on the Home Network.

As can be seen from the ACID console picture below, the source port is 31337, a well known port for it's association with hacker tools, including "BackOrifice". The destination port is 515, which is the Line Printer Daemon port. All Q related attacks have ACK and RST bits set with TCP sequence number, acknowledgement number and window size set to 0. The payload has 'cko'.



IP

source addr	dest addr	Ver	Hdr Len	TOS	length	ID	flags	offset	TTL	chksum
255.255.255.255	207.166. X.Y	4	5	0	43	0	0	0	15	15586

FQDN	Source Name	Dest. Name
	255.255.255.255	207.166. X.Y

Options	none

TCP

source port	dest port	R1	R0	URG	ACK	PSH	RST	SYN	FIN	seq #	ack	offset	res	window	urp	chksum
31337	515				X		X			0	0	5	0	0	0	61961

Options	none

Payload

```
length = 3
000 : 63 6B 6F                                             cko
```

Observed aspects from the alert packets:	
Source IP Address:	255.255.255.255
Source Port:	31337
Destination Port:	515
TCP Flags set:	ACK, RST
Sequence Number:	0
Acknowledgement Nu:	0
TTL:	15
Payload:	cko

Table 2-14: Packet Details

According to Q distribution README file, Q is 'a Client/Server suite designed to provide a maximum of security, especially confidentiality and anonymity'. The latest version of Q v2.4 uses strong RSA/libiSSL encryption for session authentication and data exchange. The server part is a program 'qd' capable of running as a standalone server or as a raw IP server. The client program 'q' has many options to connect to the server. The server can be controlled remotely from the client, but the payload is not encrypted indicating an older version of Q client in use.

2.3.5 Attack Mechanism

Below are some of the options as mentioned in the Q distribution README file. Q as a standalone server can listen on any port to serve incoming connections.

```
# qd –p 12345
```

The above command binds an encrypted shell to port 12345.

Use client ‘q’ to connect : q -l 12345 Server_IP_Address
By default the source IP address is random.

In rawIP mode, simply run the qd on the server. (Not all systems support raw IP mode). This also needs root or administrator privileges.

```
# qd    - Start the Server
Use client ‘q’ to connect: q -S -l 1234  (Or any other port)
```

In this mode the server is controlled externally by the client to spawn shells. In the above example the server spawns a shell on port 1234 for the client to connect to.
The local client source port can be set using the –s (lower case) option. Option –t allowsTRANSD mode, kind of ssh tunneling using port forwarding, allowing other local clients to connect to the client and communicating to the server via client encrypted session.

Bouncer/Relay option: The server can be run as a relay box bouncing all connections to an entirely different box..

```
# qd  -b  12345:6667:someirc.server.com
```

In this the server binds an encrypted bouncer to port 12345 and all connections will be relayed to port 6667 on server someirc.server.com.
There are many client options to send control data to the server to spawn shells, to specify protocols, to execute commands etc. More than one server can be specified in the target while executing commands.

Raw IP option can actually avoid the IDS sensors. In this mode, rather than creating a standard listener and using a standard client/server configuration the “Q“ Trojan looks for raw IP packets that match a specific parameter. The data of these packets will contain encrypted instructions that the “Q“ daemon will perform.

All these features make Q a great tool for backdoor operations. And the new encryption feature makes it very difficult to see the payload.
As seen from the above Q options it is possible to run the server on any port and send client connections and commands from any port and address. So a compromised system can be directed to listen on an allowed port like HTTP (80) and the control and data commands can be sent. Most sites allow incoming traffic from the site’s public DNS servers. So it is even possible to spoof the sites public DNS source addresses with source port set to port DNS (53).

There have been advisories for port 515, on which Unix LPR service runs. The increase in probes may make one wonder if this is all about exploiting vulnerabilities in LPR services running on port 515. But as seen from other discussion groups there have been similar attacks on other ports like HTTP (80).

2.3.6 Correlation

Many attacks with the same characteristics were found in the log files from 2002.10.1 to 2002.10.17.

As posted at http://www.securityfocus.com/archive/96/311300/2003-02-08/2003-02-14/0 by Jon 'the sensor never sees any other traffic from the source. A handful of machines (from internet) do some innocent web browsing of machines on the networks belonging to (HOME_NETWORK), and then terminate the connection. Seconds later, the 'cko' packet shows up from that host. Other times, a host on my network browses a remote site, and eventually terminates the connection. Seconds later, the 'cko' packet shows up on my doorstep from the remote site'.

As tested by Les Gordon, http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc Page 15-16, the signature development for Q is not going to be simple.

Craig Baltes discusses his test results for Q at http://archives.neohapsis.com/archives/vuln-dev/2002-q4/0070.html

Christ J Clark suggests a broken tool at http://lists.jammed.com/incidents/2001/07/0023.html

2.3.7 Evidence of Active Targeting

Was this IP address targeted at some point?
There is lot of traffic with same characteristics going to a large number of IP addresses with in the HOME_NETWORK. It seems like a reconnaissance attempt to provoke response from already compromised systems. This is not active targeting. It is also possible that there are similar attacks to other ports and proper rules are not in place to capture the traffic. Out of the randomness of the attack, it is possible the source address generated happened to be 255.255.255.255, some times, obviously not random enough!

Was this a Stimulus or Response?
This is certainly a stimulus. But could this be coming from another compromised machine? That is very highly possible. It is possible that a random class C network is being scanned for Q servers. At the end it is not very clear if this alert is because of a broken tool, bad TCP/IP stack or a real attack. There have been many instances of this attack on different discussion groups. Donald Smith suggests that this is a false positive as shown in the section 'Excerpts from detect posted to intrusions discussion group' at the end of this assignment.

2.3.8 Severity

Severity = 4

The following formula has been used to calculate severity.

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

(3+5) – (2+2) = 4

Criticality	3	The purpose of this specific system is unknown. A middle ground is taken.
Lethality	5	A successful compromise of the system will give remote access to pretty much everything.
System Countermeasures	2	Unknown. Assuming it to be minimal.
Network Countermeasures	2	Unknown. Assuming it to be minimal.

Table 2-15: Severity Calculation Table

* Each value is ranked on a scale of 1-5, 1being the least and 5 the highest.

2.3.9 Defensive Recommendation

- Review the router Access Control Lists (ACL) and firewall configurations. In general, routers should allow inbound packets only for connections established from inside. If needed, set specific filters for other public servers like WWW, SMTP to allow incoming traffic. This would ensure that the connection initiation from outside to non-public servers is stopped.
- Cisco provides a feature called "unicast reverse path forwarding" checking. It is borrowed from the IP multicast world. When Unicast RPF is enabled on an interface, the router examines all packets received as input on that interface to make sure that the source address and source interface appear in the routing table and match the interface on which the packet was received. For more information read: http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/uni_rpf.htm
- Set host based detection like tripwire and auditing on systems that directly accept incoming connections from outside.
- Gather tcpdump statistics for the public servers.
- Setup intrusion detection sensors in the home network to test traffic coming through the firewall.
  - Set specific filters to deny access to unwanted public servers like IRC servers.

2.3.10 Multiple Choice Test Question

Q) What is the typical cisco router access list command set to defend a network from being exploited by outside network employing TCP to attack non-public ports where the only allowed port from outside is 80? ($HOME_WEB_SERVER is the http server. And all communication from inside to outside is allowed.)

A) access-list 101 permit tcp any $HOME_WEB_SERVER eq http
B) access-list 101 permit tcp any any established
C) access-list 101 deny any any log
D) All of the above in the order A, B, C

Answer: **D**

Q) What is the purpose of choice 'B' in the previous question (Choose the best answer)?

A) Allow all TCP communication from anywhere to anywhere
B) Allow all TCP communication from outside
C) Allow all TCP communication from inside
D) Allow all TCP communication from outside related to the TCP connections established from inside

Answer: **D**

2.3.11 Excerpts from detect posted to intrusions discussion group:

Posted on 6/13/2003 10:55 PM at

> From: Sai Prasad Kesavamatham
> To: intrusions@incidents.org
> Sent: 6/13/2003 10:55 PM
> Subject: LOGS: GIAC GCIA Version 3.3 Practical Detect(s)

Detect 3 from assignment II
-----Original Message-----
http://cert.uni-stuttgart.de/archive/intrusions/2003/06/msg00166.html

------Comment-http://cert.uni-stuttgart.de/archive/intrusions/2003/06/msg00167.html
From: "Smith, Donald" <Donald.Smith@qwest.com>
To: "'Sai Prasad Kesavamatham '" <saik@presicorp.com>; <intrusions@incidents.org>
Sent: Saturday, June 14, 2003 2:56 AM
Subject: RE: LOGS: GIAC GCIA Version 3.3 Practical Detect(s)
Why would a hacker using a stealthy back door program like Q use a broadcast address and two well known "hacker related" ports?
-------------------Author's Response----------------------
As I mentioned in 'evidence of active targeting':
"It seems like a reconnaissance attempt...Out of the randomness of the attack, it is possible the source address generated happened to be 255.255.255.255".
I would like to clarify here that:
Q is compiled as a client/server set. Each compiled version of client and server will have a unique authentication number. To communicate with a different server, the client has to be recompiled using the authentication number used while compiling the other server. This means a client compiled with one server set can not be used with another server unless recompiled for the new server.
Some of the reasons why these particular ports and addresses are being used could be -
It is possible that the source and destination IPs and ports could have been generated in random using some other simple script to use with Q client. The IDS picked up this traffic since this traffic matched with these specific signatures. And as mentioned in the correlation, Les Gordon, described in his practical why it is difficult to create a foolproof signature for this attack.
It is also possible that a compiled Q client/server set with these 'source IP/port - destination IP/port' characteristics is distributed on the net and some one who got the client piece is trying to find the Q servers.
It is highly likely that a larger percentage of Q attacks are being overlooked because of lack of proper IDS signatures for this attack.
–sai
-------------------End of Author's Response----------------------
------Comment-http://cert.uni-stuttgart.de/archive/intrusions/2003/06/msg00172.html
From: "rocker atschool" <starplanet1000@yahoo.com.hk>

To: "Sai Prasad Kesavamatham" <saik@presicorp.com>; "Smith, Donald" <Donald.Smith@qwest.com>; <intrusions@incidents.org>
Sent: Sunday, June 15, 2003 12:33 AM
Subject: Re: LOGS: GIAC GCIA Version 3.3 Practical Detect(s)
Sai,
255.255.255.0/24 is not CLASS C network. pls check RFC1918. Since most of well-configured router will not route broadcast packet, where this broadcast comes from ?
-------------------Author's Response----------------------
http://cert.uni-stuttgart.de/archive/intrusions/2003/06/msg00179.html
Rocker,
I wanted to mention /24 as a class C network. I will change the wording.
Thank you.
As far as your second question regarding 255.255.255.255 being allowed, remember this is source address we are talking about. By default routers do forward broadcasts, meaning, when the destination address is set to 255.255.255.255.
And by default, a router (atleast Cisco) does not look at source addresses and hence a packet with source address set to 255.255.255.255 will be forwarded. ISPs are supposed to implement a protection mechanism to prevent IP spoofing. Cisco provides a feature called "unicast reverse path forwarding" checking. It is borrowed from the IP multicast world. Here is a link on this unicast reverse path forwarding:
http://www.cisco.com/univercd/cc/tc/doc/product/software/ios120/12cgcr/secur_r/srprt6/srrpf.htm
–sai
-------------------End of Author's Response----------------------
------Comment-http://cert.uni-stuttgart.de/archive/intrusions/2003/06/msg00176.html
From: "Smith, Donald" <Donald.Smith@qwest.com>
To: "'Sai Prasad Kesavamatham '" <saik@presicorp.com>; <intrusions@incidents.org>
Sent: Saturday, June 16, 2003 08:32
Subject: RE: LOGS: GIAC GCIA Version 3.3 Practical Detect(s)
I suggest you consider the possibility of this being a false positive. Look at the signature for Q and the packets. Perhaps get some Q packets and compare them to what you have here. There are some publicly available Q packets on the net. As for routers not routing 255.255.255.255 that is true as you stated. That is also a good reason to believe this is a spoofed from address (as you stated). Most routers are NOT configured to look at the from address. They only route based on the source address. There are methods that can be used to restrict bogus from addresses on routers. So please consider them as defensive recommendations.

Table 2-16: Incidents.Org discussion

2.4 References:

Snort ID Database: http://www.snort.org/snort-db/sid.html
http://www.uniras.gov.uk/sql.htm
http://securityresponse.symantec.com/avcenter/Analysis-SQLExp.pdf

3. Assignment 3: Analyze This!

This portion of the practical is a security audit for a University. The aim of the audit is to analyze the intrusion detection logs from the university and prepare a report. Information is not available on the intrusion detection used to collect these alerts. The logs were downloaded from http://www.incidents.org/logs. The log data from five consecutive days, May 16th to 20th 2003, was analyzed.

3.1 Executive Summary

The availability of past logs on the internet makes it easy for the hackers to map the internal network. There seems to be an improvement in the overall filtering process as some of the alerts like SMB did not show any corresponding outgoing traffic as pointed out by past GCIA papers.

The machines involved in IRC, red worm and tftp related activity need special mention. The link graphs in Fig. 6 and 7 show a part of the affected network. The analysis in section 3.4.2 clearly indicates that the red worm detection rules are very generic and could potentially hide the real threats under high noise levels from false positives complicating the analysis process. This is also true with some other detection rule sets as shown in later sections.

TFTP alerts mostly involved machines participating in peer-to-peer activity like games, file sharing etc. This allows for easy infection of internal network.

MY.NET.97.0 network is involved multiple scans and the following alerts. As seen in some other client locations, there is a possibility that this network belongs to a VPN subnet or a dial-up subnet. Proper access control lists and other security measures are needed to contain such a network. One way is not to use split tunnel in VPN configurations ensuring all the traffic goes only through the university network. But this would also raise many detect alerts since an already compromised dialup or vpn client tries to contact the peers through the university network.

Internal machines listed in table 3-31 are involved in scanning activity and need to be checked and monitored.

Access-control lists on the network perimeter routers and firewalls need configuration changes as recommended under section 3-4 to filter ingress and egress traffic.

Check machines listed in table 3-31 for security compromises.
Refer to host and peer to peer analysis for compromised machines.
Refer to recommendations under sections 3.4.1.1.1, 3.4.1.2 and 3.7 for more information on network perimeter and other access list configurations.

3.2 Log Files and Formats

3.2.1 List of Files Analyzed:

Alert Files	Scan Files	Out of Specification Files
Alert.030516	Scans.030516	OOS_Report_2003_05_16_6191
Alert.030517	Scans.030517	OOS_Report_2003_05_17_14869
Alert.030518	Scans.030518	*OOS_Report_2003_05_19_9515
Alert.030519	Scans.030519	OOS_Report_2003_05_19_9542
Alert.030520	Scans.030520	OOS_Report_2003_05_20_14142

Table 3-1: Log Files used in Analyze This!

*It is assumed that this OOS file was actually for 18th since there was no other file available for this date and all entries corresponded to that date.

3.2.2 Alert Logs

The alerts are in a single line format. Each line has the following fields in the order; 'date-time stamp', 'alert message', 'source IP address', 'source port number', 'destination IP address', 'destination port number'.

```
<date-time stamp>     [**] message [**]     <SRC IP>:<SRC Port>  ->  <DST IP>:<DST Port>

05/16-00:30:05.828186 [**] SMB Name Wildcard [**] 81.49.170.82:1030 -> MY.NET.210.196:137
```

The first two octets of the internal network IP addresses were obfuscated with "MY.NET" to sanitize sensitive information.

3.2.3 Scan Logs

The scan log format is almost similar to alert log format. The differences are that the date-time stamp does not contain microseconds and the last entry of the line contains either "UDP" or "TCP" flags.

```
<date-time stamp>  <SRC IP>:<SRC Port>  ->  <DST IP>:<DST Port>  [<TCP Flags> | UDP]

May 16 00:00:06    MY.NET.196.193:3339  ->   216.154.156.27:17300     SYN   ******S*
May 16 00:01:17    MY.NET.239.158:4706  ->   212.195.128.193:8003     UDP
```

The original scan logs contained real IP addresses from university and were sanitized during analysis by replacing the first two octets of the IP address with 'MY.NET'. This sanitation was achieved using one of the Perl scripts 'parsescans.pl' shown in Appendix A.

3.2.4 OOS Logs

OOS logs are created by running Snort in sniffer mode (snort –v). These logs are in multi-line format. The first line of an entry contains the basic packet information. The remaining fields of the entry contain some packet header specifics and optionally part of the packet contents.

```
<date-time stamp>       <SRC IP>:<SRC Port>  ->  <DST IP>:<DST Port>
```

Log sample:

```
=+=++=+=+=+=+=+=++=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/15-00:05:37.245156 80.55.53.78:33763 -> MY.NET.201.130:6346
TCP TTL:50 TOS:0x0 ID:41585 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x9ADF7626  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 66573698 0 NOP WS: 0
=+=++=+=+=+=+=+=++=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

3.3 Analysis Process

All the logs were parsed using Perl scripts and stored in MySql database. Basic SQL commands have been used to analyze the data. Some previous GCIA practical papers were reviewed for Perl script code ideas.

It is found that some of the lines in the 'alert.*' log files are concatenated and a few lines have only a port or IP & port fields, eg:1542 -> 192.168.10.1:1542 and others have additional unwanted information. These lines have been neglected for the analysis.

MySql schema for logs:
A database 'logs' was created with three tables one each for alerts, scans and oos. The fields used for different log tables are shown below.

Table ALERTS:
time datetime, mseconds int, alertsign varchar(50),srcip varchar(25),srcport int,dstip varchar(25),dstport int with one additional sid field for auto-increment.
Table SCANS:
time datetime, byhour int, scansign varchar(10),srcip varchar(25),srcport int,dstip varchar(25),dstport int, protocol varchar(8),flag varchar(10) with one additional sid field for auto-increment.
Table OOS:
Create table oos (sid int not null auto_increment, time datetime, byhour int, mseconds int, oossign varchar(10),srcip varchar(25),srcport int,dstip varchar(25),dstport int with one additional sid field for auto-increment.

For easy maintenance during test, a text file 'createlogsdatabase.txt' was created. The database was created using the following command from command line interface.

```
# mysql < createlogsdatabase.txt

### content of createlogsdatabase.txt###
drop database logs;
create database logs;
connect logs;
create table alerts
(sid int not null auto_increment, time datetime, mseconds int, alertsign
varchar(50),srcip varchar(25),srcport int,dstip varchar(25),dstport int, primary
key(sid));

create table oos
(sid int not null auto_increment, time datetime, mseconds int, oossign
varchar(10),srcip varchar(25),srcport int,dstip varchar(25),dstport int, primary
key(sid));

create table scans
```

```
(sid int not null auto_increment, time datetime, scansign varchar(10),srcip
varchar(25),srcport int,dstip varchar(25),dstport int, protocol varchar(8),flag
varchar(10),primary key(sid));
```

Table 3-2: Schema for logs - MySQL database

Three Perl scripts, 'parsealerts.pl', 'parsescans.pl' and 'parseoos.pl' have been used to parse the log files and insert the data into MySql database. The scripts are shown in appendix A. Some of the SQL commands used are also shown in Appendix A. An additional Perl script 'common.pl' (shown in appendix A) has been used to find all the unique common IP addresses between different tables in the database. Some of the database operations for finding the common IP addresses among 'alerts' and 'scans' logs, particularly for destination IP addresses, took a very long time resulting in time outs. The Perl script 'common.pl' was used in that situation.
The advantage with the backend database procedure is that at any time new data can be appended or the data can be reset to null. Different queries can be generated on the database. Some of the SQL commands used in the analysis are shown in Appendix A.

3.3.1 Analysis Method

Three lists have been created from alert logs showing all alerts (table 3-3), possible internal hosts and services (table 3-4) and hosts participating in peer-to-peer networks (table 3-6). For example alerts showing 'external web traffic' indicate web servers. Similarly analyzing the ports, involved in alerts showing 'Notify Brian B. 3.56 tcp' may give further indication of a special network.

A list of top talkers was created for certain alerts to help the analysis. Common IP addresses (addresses appearing in more than one type of log) categorized by source and destination IP addresses from alerts, scans and oos logs have been analyzed. For example an external IP address appearing in both alerts and scans log file has a special significance in the sense that it is probably more than just a coincidence. Similarly an internal address appearing as a source IP address in scans and in alerts may indicate a compromised system.

The scan and oos logs have been analyzed in an effort to correlate analysis from alerts. A list of internal IP addresses scanning internal IP addresses has been created. Port scanning originating from internal network indicates either compromised hosts or malicious activity from internal network or scanning by internal security groups.

There is a small error margin involved in the statistics generated because of the file format errors and parsing process involved.

3.4 Alerts Analysis

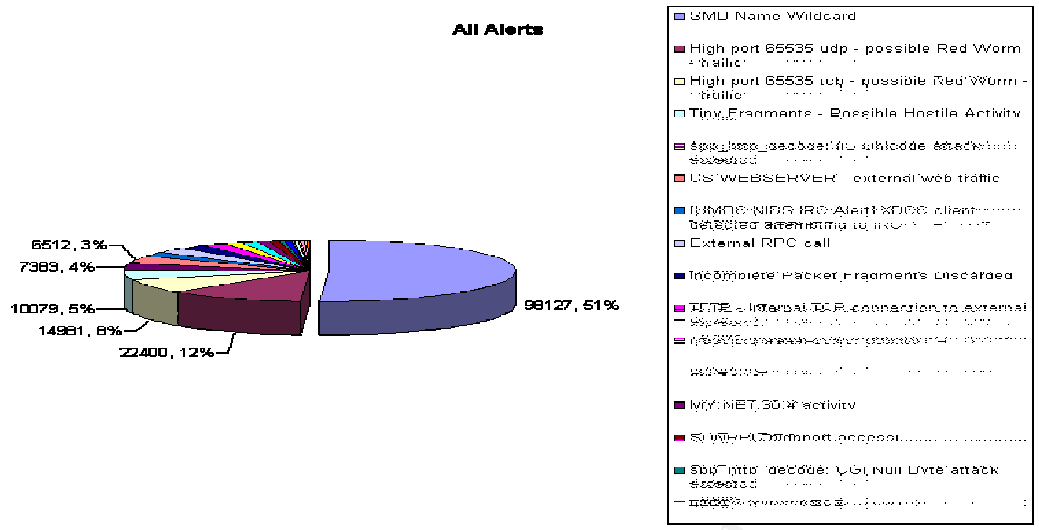


Figure 5: Graphical Representation of All Alerts

Total number of alerts for five days is 193039. Table 3-3 below shows all the alerts and the number of times they occurred during the five days of analysis.

Alert Message	# Total
SMB Name Wildcard	98127
High port 65535 udp – possible Red Worm – traffic	22400
High port 65535 tcp – possible Red Worm – traffic	14981
Tiny Fragments – Possible Hostile Activity	10079
Spp_http_decode: IIS Unicode attack detected	7383
CS WEBSERVER – external web traffic	6512
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	5359
External RPC call	5099
Incomplete Packet Fragments Discarded	4479
TFTP – Internal TCP connection to external tftp server	3949
Possible trojan server activity	2943
Null scan!	2501
MY.NET.30.4 activity	2393
SUNRPC highport access!	1445
Spp_http_decode: CGI Null Byte attack detected	1251
Queso fingerprint	877
IDS552/web-iis_IIS ISAPI Overflow ida nosize	488
TCP SRC and DST outside network	378
EXPLOIT x86 NOOP	374
CS WEBSERVER – external ftp traffic	369
MY.NET.30.3 activity	242
[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.	228
IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize	208
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	134
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected	114

attempting to IRC	
SNMP public access	108
IRC evil – running XDCC	99
NIMDA – Attempt to execute cmd from campus host	75
NMAP TCP ping!	73
Connect to 515 from outside	65
TFTP – Internal UDP connection to external tftp server	63
EXPLOIT x86 setuid 0	60
EXPLOIT x86 stealth noop	32
EXPLOIT x86 setgid 0	30
FTP DoS ftpd globbing	15
Probable NMAP fingerprint attempt	14
Notify Brian B. 3.56 tcp	14
Notify Brian B. 3.54 tcp	12
EXPLOIT identd overflow	12
EXPLOIT NTPDX buffer overflow	7
[UMBC NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot	5
[UMBC NIDS IRC Alert] K:line"d user detected, possible trojan.	5
SYN-FIN scan!	5
TFTP – External UDP connection to internal tftp server	5
SMB C access	4
DDOS shaft client to handler	4
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	4
Attempted Sun RPC high port access	3
Back Orifice	3
DDOS mstream client to handler	2
EXPLOIT x86 NOPS	2
NETBIOS NT NULL session	1
DDOS TFN client command BE	1
RFB – Possible WinVNC – 010708-1	1
External FTP to HelpDesk MY.NET.83.197	1
External FTP to HelpDesk MY.NET.53.29	1

Table 3-3: All Alerts

As of 05/31/2003 the top ranked attacks reported at http://isc.incidents.org are Microsoft-ds (port 445), NetBIOS-ns (port 137), www (port 80), ms-sql-m (port 1434) and NetBIOS-ssn (port 139). It is important not to get carried away by the amount of alerts from the university logs. Similarly, the top-talkers list from tables below is to give an overall picture. Some real attacks may get overlooked if the noisy unwanted alerts are not properly filtered.

3.4.1.1 Host Table Mapping for Internal Network

The idea behind constructing a host table map is to recognize and isolate the server services from other non-server networked hosts or client hosts in the hope to categorize the noisy traffic. Table 3-4 below lists some of the hosts and the possible services that are being served from these hosts.

Hosts	Port-Service	Remarks
MY.NET.3.54; MY.NET.3.56; MY.NET.30.3; MY.NET.53.29; MY.NET.83.197; MY.NET.100.165; MY.NET.212.90	21 – FTP	From alerts to Help Desk & external traffic. Most of the services were also found from alert to Brian
MY.NET.60.11; MY.NET.60.39	22 – SSH	
MY.NET.1.4; MY.NET.179.78; MY.NET.237.22	23 – TELNET	
MY.NET.100.230; MY.NET.6.34; MY.NET.6.40, MY.NET.6.47, MY.NET.MY.NET.179.78; MY.NET.25.11; MY.NET.24.21; MY.NET.24.22; MY.NET.24.23 MY.NET.12.2	25 – SMTP	A large percentage of the alerts on port 25 came from 12.165.28.10 (tcp/65535) an AT&T cable network IP address. And there were scans from the same IP address from port 65535 to port 25 in the university network.
MY.NET.1.3; MY.NET.1.4; MY.NET.1.5; MY.NET.137.7;	53 – DNS	

MY.NET.30.4; MY.NET.100.165; MY.NET.5.15;
 	
MY.NET.5.0; MY.NET.29.0; MY.NET.150.0; MY.NET.130.0; MY.NET.225.58; MY.NET.252.142; MY.NET.137.18; MY.NET.111.21; MY.NET.110.224; MY.NET.112.216; MY.NET.106.222 ; MY.NET.233.146; MY.NET.86.19; MY.NET.235.70;
	80 – HTTP	External Web Traffic & Alerts to Brian.
		Others are found from x86 exploit alerts. MY.NET.5.0 and 29.0 seem to have many web application servers.

MY.NET.12.4	110 – POP	
MY.NET.186.135; MY.NET.247.71; MY.NET.87.16; MY.NET.99.253; MY.NET.233.83	111 – RPC	Mostly from 140.121.175.75 ministry of education Taiwan.
MY.NET.24.8	119 – NNTP	News Server Traffic from Exploit x86 alerts
MY.NET.12.2	143/993 (IMAP& IMAP SSL)	OOS logs

MY.NET.190.94; MY.NET.190.93; MY.NET.190.100; MY.NET.190.19	135, 137, 445 – NetBIOS related	SMB C access is normally from an established connection to windows server port 137. Alert source 80.37.21.67 appeared in dshield.
MY.NET.24.15	515 – LPD (Print Server)	

Table 3-4: Internal Hosts Map - created from alert logs

* Some of the hosts shown above may be co-hosting other services.

3.4.1.1.1 Some conclusions from host table entries in table 3-4 and related logs

SMTP:
All the alerts to port 25 came from three outside sources and appeared under 65535 (tcp) red worm traffic using source port 65535. 983 attacks out of 991 came from 12.165.28.10. Address 12.165.28.10 belongs to a cable modem network. And the scans from 12.165.28.10 during this time indicate that 12.165.28.10 is possibly a compromised host and does not seem to be a valid SMTP server.

Some of the internal hosts in the alert logs using source port 25 were mapped as SMTP hosts shown in table 3-4 above. Many internal sources communicated with 12.165.28.10 using source port 25 and destination port 65535. Interestingly there is only one instance of established connection as seen below from alerts log file.

```
05/16-21:58:37.355050  [**] High port 65535 tcp - possible Red Worm - traffic [**]
12.165.28.10:65535 -> MY.NET.98.104:25

05/16-21:58:37.523518  [**] High port 65535 tcp - possible Red Worm - traffic [**]
MY.NET.98.104:25 -> 12.165.28.10:65535
```

No reason is known why host MY.NET.98.104 is not involved in more alerts during the five days of analysis if it has already been compromised. Obviously the long list of internal hosts using source port 25 may not truly represent SMTP servers. A few of them are infected and probably compromised by red worm. Red worm link graph is discussed later under IRC alerts.

Recommendations:

- Disable mail relay for every one. Use spam filters. Use anti-virus software to detect and clean viruses at the mail gateway.

DNS:
DNS servers got lot of NMAP and SNMP public access alerts. These servers need to be protected well. It is very easy to map internal host information by querying the name server. Just limiting zone transfers to certain hosts and feeling secure about it is only false security. Limiting zone transfers may effectively block domain listing commands like running ‘ls <domain name>’ under dns-nslookup to list the entries in the zone file. But the internal network host table can be mapped

by sending reverse lookup queries. Since most sites try to use host names that are relevant to the application they are serving, it is possible to map internal IP addresses and services.

Recommendations:

- Set proper access control lists in place to block all DNS queries from external networks to internal DNS servers.
- If possible use split DNS to isolate external DNS servers from internal DNS servers. There is absolutely no need for outside world to resolve internal name space.
- Disable SNMP if not absolutely needed. If not, set proper access control lists to restrict SNMP traffic.
- Update relevant OS and BIND patches.

HTTP:
Networks MY.NET.5.0 and MY.NET.29.0 showed up in Exploit 86 alerts. Hosts from these networks used source port 80 indicating that either the hosts from these networks are heavily populated with web servers (most probably Microsoft IIS) or compromised to use port 80 as source port. Proper filtering and fine-tuning of these networks would reduce the number of false positives from exploit x86 rule sets.

POP: Targeted activity was found from 140.121.175.75 and 151.192.238.93. Internal host MY.NET.12.4 communicated on ports 110 (POP) and 993 (IMAP-SSL) with cable network clients. It is possible that the outside clients are genuine clients communicating occasionally using port 65535. This is most probably a false positive. As source, MY.NET.21.48 was involved in sending external RPC calls to 140.121.175.75 (belonging to Taiwan – Ministry of Education) and external port 80 access to 211.95.129.136 'IDS552/web-iis_IIS ISAPI overflow' alerts.

The IP address 140.121.175.75 (Taiwan – Ministry of Education) is also involved in RPC alerts heavily. Address 140.121.175.75 appeared in dshield database, in attacks against ports 80 and 111. On first observation it appears as if this could be regular traffic between two university departments. But on further analysis under RPC it appears the IP address 140.121.175.75 is involved in other scanning activity suggesting targeted activity. Host MY.NET.25.11 received large number of NULL scans from 151.196.238.98 as well as other alerts. Overall there are no major concerns.

RPC:
Most of the communication is from 140.121.175.75 (Taiwan – Ministry of education). This address also appeared in SMB alerts. And this address appeared in scans for port 111. This host is trying to find open RPC ports. This is targeted activity.

Recommendations:

- Make sure RPC traffic to port 111 is blocked from outside.

- If this is a needed communication, modify the snort alert rule to minimize the false positives.

Register a complaint with the concerned authorities.

LPD (Print Servers):
Sixty-five attempts were found from one IP address 68.54.94.58 (source port 679) to MY.NET.24.15 (destination port 515, LPD) during May 18th between 19:41 and 20.57. No other alerts were found in 'scans' or 'oos' logs from these addresses. Interestingly port 679 is used by MRM (Multicast Routing Monitor) protocol. The description is available at http://www.ietf.org/proceedings/99nov/I-D/draft-ietf-mboned-mrm-use-00.txt. In a nutshell, MRM managers can instruct a receiver host to monitor multicast traffic on a specified group of addresses. This needs further analysis since even if the source address involved is one of the faculty members', there is no need to monitor from home network VPN. The source IP 68.54.94.58 is from Comcast cable network, Baltimore.

While checking the alerts from Brian's network activity, an attempt to port 4000 from IP address 62.253.140.193 was found. A check for 62.253.140.193 in dshield network showed this IP address as an attacker with 49,698 hits. Port 4000 is an ICQ outgoing port as well as home for skydance Trojan.

The following addresses were found in the dshield database (http://www.dshield.org) for same port attacks.
140.121.175.75 attacking port 111 (RPC) and 66.196.72.46 attacking port 80.

```
Whois Source domain lookup
#################################
NetRange:  140.117.0.0 - 140.138.255.255
Ministry of Education Computer Center TANET-BNETA (NET-140-117-0-0-1)
National Taiwan Ocean University TANET-B-NTOU (NET-140-121-0-0-1)
#################################
NetRange:  140.121.0.0 - 140.121.255.255
Ministry of Education Computer Center
National Taiwan Ocean University
#################################
 NetRange:   66.196.64.0 – 66.196.127.255
CIDR:       66.196.64.0/18
NetName:    INKTOMI-BLK-3
```

Table 3-5: Who is database search results

3.4.1.2 Map of Peer-to-Peer Users

Table 3-6 below shows the hosts in the university network participating in peer-to-peer applications like napster and kazaa.

Gnutella Peers	Destination Port	Napster	Destination Port
MY.NET.194.223, MY.NET.249.122,	6346	MY.NET.217.134 MY.NET.100.165	6699

MY.NET.210.142		MY.NET.202.206	

Neo-Modus Peers
 	
MY.NET.236.94
MY.NET.237.62
MY.NET.237.62
MY.NET.237.62
MY.NET.202.18
MY.NET.202.18
MY.NET.221.242
	412	**Kazaa**
		
		MY.NET.209.78;
		MY.NET.196.159;
		MY.NET.194.13;
		MY.NET.240.114;
		MY.NET.217.90
			1214

WinMX File sharing MY.NET.250.162*	6257 (*Top Talker from scans)		

Table 3-6: Peer-to-Peer client map

Analysis of entries from the host table and peer-to-peer network maps also showed a large number of red worm and SMB alerts. The SMB alerts did not get response from inside network. This is further strengthened by the SMB alert analysis in later sections.

Recommendations:

- Special ACL rules have to be set on the networks with peer-to-peer channels, which seem to have student computers, to monitor and block unwanted traffic. The networks with peer-to-peer machines are a threat to the network and system integrity and are vulnerable. These systems are also stealing bandwidth. Check the systems for any backdoor Trojans.

3.4.2 Top Talkers from Alert Logs

Table 3-7 below shows the top-ten source and destination addresses from the five day alert logs. The highlighted entries represent the top-talkers from university's internal network.

Top Ten Sources		**Top Ten Destinations**	
Source IP	**# Total**	**Destination IP**	**# Total**
MY.NET.235.110	9783	MY.NET.100.165	6957
MY.NET.201.58	9544	MY.NET.201.58	6314
140.121.175.75	5099	205.188.149.12	4630
MY.NET.198.221	4635	MY.NET.153.157	4008
140.142.19.69	3943	MY.NET.202.206	3640
12.235.36.52	3313	66.67.29.21	2482
MY.NET.251.10	2736	MY.NET.30.4	2392
MY.NET.202.206	2061	24.157.3.20	2042
208.45.250.203	1883	MY.NET.251.10	1915
172.17.3.17	1873	12.235.36.52	1858

Table 3-7: Top Talkers from Alerts - Source & Destination Addresses 🟦 **Home Network**

Table 3-8 below shows the top-ten source and destination ports from the five day alert logs.

Top Ten Source Ports		**Top Ten Destination Ports**	
Source Port	**# Total**	**Destination Port**	**# Total**
65535	21835	137 (NetBIOS)	97987
0	16984	0 (Illegal)	17202
137	13548	80 (HTTP)	17178
1025	10869	65535 (High Port)	15511
1026	10484	5121(NWNGame)	12157
1027	9589	6667 (IRC)	5413
5121	8133	111 (RPC)	5090
1028	8013	4116	3296
1029	6309	27374 (Trojans)	2438
1030	2902	69 (TFTP)	2107

Table 3-8: Top Talkers from Alerts - Source & Destination Ports

Here is an example of how numbers can mislead analysis on the first look. Compare red worm alerts from table 3-3 with the numbers for destination port 5121 in table 3-8. Total number of red worm alerts from table 3-3 is 37381. The Snort signature for red worm looked for the presence of port 65535 in source or destination port. Leaving the fact aside that port 65535 is a valid ephemeral port that can be used by any application, the total number of alerts from table 3-8 for destination ports 65535 and 5121 comes to 27667 and total number of alerts for all 5121 ports comes to 20290. Port 5121 is used by a game 'Neverwinter Nights' from http://www.bioware.com. But all port 5121 alerts appeared under red worm alert since the detection rules do not have any signatures for port 5121. As seen here the false positives for red worm alert are high making it difficult to analyze the real alerts. The traffic to port 5121 is probably game traffic that just happened to use port 65535 on the other side.

But as seen under XDCC alert analysis there are some internal machines that got affected by red worm.
Recommendations:

- The red worm rule is very generic and results in too many false positives. This rule needs fine-tuning. Similarly add a new rule to monitor port 5121. Allowing machines to participate in internet games is a potential problem for future compromises.

3.4.3 Alerts of interest

Some of the interesting alerts from table 3-3 are analyzed here.

3.4.3.1 SMB Name Wildcard

Traffic Flow: Incoming - 97982 Outgoing – 0 Classification: Noise
Unique hosts from outside to inside: 97982

Unique hosts from outside to outside: 135

NetBIOS name lookups going to port 137 raised this alert. This is a standard NetBIOS traffic 'nbstat' to elicit a node status response from NetBIOS and Samba clients. A simple command like 'nbtstat –A' can be used to gather information. This is expected traffic on a Microsoft Windows Network. But it is not necessary to implement this signature rule to monitor internal to internal traffic. (Ref. http://www.sans.org/resources/idfaq/port_137.php)

Out of the 98127 of these alerts 13545 alerts had both the source port and destination port as 137. And all source IP addresses were from outside. The reasons for this could be either the alert signature is just looking for incoming traffic or no traffic is found going out. There could be new filtering in place, for traffic on port 137, after some of the past GCIA practical papers reported security compromise for this port. This traffic is classified as noise since there is no evidence to show any related packets leaving the internal network.

Similarly 4 alerts were found for 'SMB C access' to destination port 139. Port 139 is used for NetBIOS file and print sharing on windows machines. All the 4 destination IP addresses match to the Windows Servers shown in the host map table. MY.NET.190.94, MY.NET.190.93, MY.NET.190.100 and MY.NET.190.19.

Out of the four source IP addresses 80.37.21.67 appeared as an attacker in the dshield database attacking port 137.

The 'who is' lookup for address 80.37.21.67 on http://www.arin.net OrgName: RIPE Network Coordination Centre; OrgID: Address: Singel 258;Address: 1016 AB;City: Amsterdam;Country: NL The only SMB Null session alert did not ring any bells with other log files. There were no entries for any these IP addresses in the scans and OOS logs.

Table 3-9: Arin database search results

Table 3-10 below shows the top-ten source and destination addresses for SMB name wild card alerts from the five-day logs.

Sources	# Total	Destinations	# Total
208.45.250.203	1883	MY.NET.12.2	734
172.17.3.17	1873	MY.NET.24.34	643
61.132.74.72	1823	MY.NET.194.13	471
213.175.62.253	1441	MY.NET.211.110	434
69.20.128.92	1374	MY.NET.29.11	282
66.57.217.8	637	MY.NET.24.44	253
209.103.204.76	214	MY.NET.206.102	228
80.202.37.208	173	MY.NET.29.3	218
213.10.104.41	119	MY.NET.12.4	203
151.196.110.47	108	MY.NET.24.58	154

Table 3-10: Top SMB Talkers

Recommendations:

- Enable firewall filtering to block SMB related traffic. This signature can be disabled from external to internal once proper firewall rules are in place to help reduce the noise levels and false positives.
- The Windows servers shown in the host map table have to be monitored for any attacks. Because of the availability of past logs on the Internet it is possible to gather information on the internal network. Overall there have been no alerts for outgoing traffic. It is a good sign.

3.4.3.2 Tiny Fragments – Possible Hostile Activity

Traffic Flow: Incoming – 93 Outgoing – 9786 Classification: Noise
Unique hosts: Incoming – 14 Outgoing - MY.NET.235.110

The breaking up of a single IP datagram into two or more IP datagrams of smaller size is called IP fragmentation. Transmission mediums have a limit on the maximum size of a frame (MTU – Maximum Transfer Unit) that can be transmitted. The design of IP accommodates MTU differences by allowing routers to fragment IP datagrams as necessary. The receiving station is responsible for reassembling the fragments back into the original full size IP datagram. IP fragmentation can also be used for malicious purposes to avoid being recognized by firewalls and intrusion detection systems.

All the tiny fragment traffic is directed at one internal IP address MY.NET.235.110. This IP address is not found in any other alerts or oos/scan logs. Two destination ports 0 and 56464 have been used. Port 56464 is associated with beacon (client-to-client) multicast traffic. Multicast is a way of distributing IP packets to a set of machines configured to receive the traffic. For example Multicast Beacon server from http://dast.nlanr.net/Projects/Beacon/ uses port 56464 for client-to-client multicast traffic for measuring multicast performance. Multicast traffic is also used in other applications like video conferencing. Most of the outside IP addresses involved in tiny fragment traffic showed up as registered to universities or research centers with some exceptions of cable network clients. It is possible that the university has something in place to use multicasting. But the traffic from and to port 0 is not according to the RFC standards. The tiny fragments seem to result from a misconfiguration of the routers.

This also brings up the address 233.2.171.1 for verification. Twenty-four alerts were found going to 233.2.171.1 which is a reserved address space for IANA (Internet Assigned Numbers Authority http://www.iana.org), out of which only 7 alerts were from internal network. The following internal addresses communicated with 233.2.171.1.

Source IP	Source Port
MY.NET.198.221	1026
MY.NET.201.58	1035
MY.NET.97.215	32885
MY.NET.201.58	1044

MY.NET.202.206	32857
MY.NET.235.110	32771
MY.NET.233.226	1026

Table 3-11: Internal sources talking with 233.2.171.1

Address 233.2.171.1 is associated with 'beacon-mcast.accessgrid.org'. A visit to http://www.accessgrid.org reveals that The Access Grid project supports group-to-group communication via high-speed networking providing high quality audio and real-time video. As expected this site is involved in some kind of multicasting.

MY.NET.235.110 was also communicating with other outside machines particularly using ports -32771, 33798 etc. There are some known attacks on port 32771 on older Solaris operating systems. Older sun operating systems use rcpbind sometimes on port 32771 instead of 111. Red Hat clients are also known to use this port for outside communication.
Correlation: Similar attack to ports 32772 were mentioned at http://www.health.ufl.edu/mail-archives/unix-sec/1999/09/msg00006.html.
MY.NET.235.110 is also involved as destination IP in the following alerts.

Alert - Tiny Fragments – Possible Hostile Activity with 148.137.237.121 *Alert - SMB Name Wildcard* with 81.220.224.160. Alert - **eb-iis_IIS ISAPI Overflow ida nosize** with 219.130.8.186 *Alert – Null Scan!* With 148.137.237.121

Table 3-12: Other alerts to MY.NET.235.110

Further check on the above alerts did not reveal any additional information other than some scans.

Recommendations:

- Alerts showing outside IP addresses talking with 233.2.171.1 add noise to the detection logs. Fine-tune the detection rules to drop these alerts.
- If the internal machine MY.NET.235.110 is running any software related to multicasting then set proper access control lists on the firewall to allow access to this machine.
- Make sure proper authentication scheme is in place. If possible limit the access from outside network to the required IP address ranges. Fine-tune the software and router configurations to avoid fragmentation.
- If this machine is not running any software related to the above discussion, then monitor the traffic to this server and act accordingly.

3.4.3.3 High port 65535 udp/tcp – possible Red Worm – traffic

This alert is for traffic going to UDP and TCP port 65535. The numbers indicate a high volume of possible red worm traffic. The network is infected by red worm tough there are high number of false positives because of other peer-to-peer

traffic. Refer to the analysis under top destination ports and link graph in Figure 7 under section 3.4.3.1.1.
Correlation: Michael Wisener's whitep paper shows peer-to-peer noise as well as infected machines.
http://www.giac.org/practical/GCIA/Michael_Wisener_GCIA.pdf

Recommendations:

- Stay current with OS and web server patches.
- Disable email preview pan in outlook email client.
- If the machine is infected, follow the instructions at http://www.cert.org/tech_tips/win-UNIX-system_compromise.html.

3.4.3.4 External RPC call

Remote Procedure Call (RPC) is a client/server infrastructure that allows an application to be distributed over multiple heterogeneous platforms. RPC allows a client component to access remote server component without knowledge of the network address or any other lower-level information.

The portmapper or rcpbind service runs typically on port 111. For example a server service (X) that wants to take advantage of portmapper does not directly choose a service port, but rather asks the "portmapper" or "rpcbind" to assign it a port. The service X then binds to the port assigned. When a client wants to find the service X it queries the portmapper and asks for the location of the RPC service it wants. The client then connects to the relevant port if the portmapper returns the relevant information. One of the security threats is that the portmapper can be persuaded to "forward" requests to services, which may then be fooled, into thinking that they come from the local machine. The standard portmapper provides very little in the way of access control or logging. Most of the external traffic is from 140.121.175.75 to internal port 111.
Table 3-13 below shows the top-ten source and destination addresses involved in external RPC calls.

Source IP	# Total	Destination IP
140.121.175.75	5099	MY.NET.200.71, MY.NET.187.127 MY.NET.161.123, MY.NET.209.53 MY.NET.18.143, MY.NET.32.115 MY.NET.87.156, MY.NET.240.193 MY.NET.147.147

Table 3-13: External RPC - Top Talkers

As discussed under host map section 3.4.1.1, this is targeted activity.

Normally, the rpcbind service only listens on port 111. Under Solaris, the rpcbind service also listens under port 32771, which sometimes allows attackers to bypass packet filtering. http://www.iss.net/security_center/static/330.php

Table 3-14 below shows the top-ten source and destination addresses for sunrpc high port access alerts.

Source IP	# Total	Destination IP	# Total
24.125.66.19	1363	MY.NET.252.78	1365
64.12.26.97	11	MY.NET.226.26	24
64.106.159.160	10	MY.NET.88.208	7
207.46.106.82	7	MY.NET.203.98	6
64.12.30.61	6	MY.NET.149.36	6
216.239.57.99	5	MY.NET.97.212	5
129.2.56.185	5	MY.NET.149.13	5
131.118.254.39	5	MY.NET.99.11	5
172.172.210.82	4	MY.NET.149.45	3
64.12.30.117	3		

Table 3-14: Sun RPC High Port Top Talkers

Recommendations:

- Stop RPC services if not needed by any applications.
- There is no need for external machines to access internal RCP port 111. Block all port 111 traffic from entering the network. Update any RCP related and system related patches.

3.4.3.5 TFTP – External to internal tftp and other tftp alerts

Total alerts: 4012   Classification: Risk
TFTP (Trivial File Transfer Protocol) is a simple form of FTP (File Transfer Protocol). TFTP uses TCP and UDP protocols and does not provide security features. It is often used by servers to boot diskless workstations and routers and to load system configurations. Table 3-15 below shows internal TFTP servers.

Outside clients	#Registered To	Internal TFTP Servers
4.62.30.53	Verizon DSL	MY.NET.196.29
218.159.24.157	Under Korean Network	MY.NET.196.29
211.49.82.138	Under Korean Network	MY.NET.196.29
172.187.79.122	AOL	MY.NET.205.130
63.250.205.60	Yahoo broadcast	MY.NET.233.122

Table 3-15: Internal TFTP Servers

Host MY.NET.196.29 appeared in 'BOT' alerts particularly in the Back Orifice alerts. This host is also targeted in many scans. On initial analysis this machine appears to be compromised. However there are no scans or other qualified alerts showing this machine as a source. The other two hosts MY.NET.205.130 and MY.NET.233.122 appear to be participating in yahoo/aol peer to peer networks uploading files. It is not clear if these uploads are user initiated or not. These machines probably belong to the student network. Read further analysis on MY.NET.196.29 under bot alerts. Address 12.211.182.115 is one of the external tftp servers involved in malicious activity. Refer to link graph in Fig. 7 for the analysis.

On the other hand network MY.NET.205.0 is involved in a large number of alerts including internal to external tftp server, possible Red Worm using TCP and UDP and IIS Unicode alerts. A quick check on one of the top IP addresses MY.NET.205.234 revealed bi-directional TFTP connections with many external hosts shown here. 62.118.251.33, 64.12.27.84, 64.12.27.87, 64.12.28.189, 64.12.28.97, 203.165.153.100, 205.188.153.12, 205.188.9.187, 210.202.65.126, 213.180.194.129, 219.179.224.134. Again most of the addresses belong to AOL. This shows these machines are participating in chat services and file uploads.

```
NetRange:   64.12.0.0 - 64.12.255.255
CIDR:       64.12.0.0/16
OrgName:    America Online, Inc.
OrgID:      AMERIC-158
Country:    US

NetRange:   205.188.0.0 - 205.188.255.255
CIDR:       205.188.0.0/16
NetName:    AOL-DTC

inetnum:      210.202.64.0 - 210.202.87.255
netname:      CCTV-HT-AP
country:      TW
descr:        Chun-Chien Cable Television Co., Ltd.
descr:        No. 159, Sec. 3, Wen-Hsing Rd., Xi-
Tun Dist.,Taichung City, Taiwan, R.O.C.
```

Table 3-16: Who is database search results

Recommendations:

- The following machines need a thorough check for backdoors and other vulnerabilities. MY.NET.91.109, MY.NET.196.73, MY.NET.203.42, MY.NET.206.210, MY.NET.217.62, MY.NET.224.242, MY.NET.236.166
- Block TCP/UDP port 69 (TFTP) traffic at ingress and egress points.

3.4.3.6 EXPLOIT x86

Classification: Noise

These exploits try remote buffer overflow attacks by sending binary shell code to take advantage of insecure coding practices. The attack may some times result in executing attacker's code of choice. The attacks may contain code to change the identity of the current group to that of the root group or current user to that of root user.

Alerts to ports below 1024 were analyzed first. The ports came up as 20(ftp), 80(http), 119(nntp), 412 and 443(ssl). Port 412 (TCP and UDP) is used by Neo Modus a peer-to-peer file sharing application. Most of the alerts were results of http connections. Some of the alerts to port 119 came from sources belonging to University of Maryland.

setgid0: The top source port is 80 (HTTP). The source addresses are genuine web sites hosting web services. It is possible that these web servers are returning content that includes strings of NOOP characters. Destination port 412 was involved in setgid0 alerts. Internal host MY.NET.202.18 was involved as source IP for CGI Null Byte alerts to 12.31.2.46. Scans were found from outside network to this IP (MY.NET.202.18) on port 412.
Setuid0: MY.NET.84.146 (4691) communicated with 61.130.174.138 (20). This is a Chinese source. Investigate further.

All x86 Exploits	**#**
EXPLOIT x86 NOOP	374
EXPLOIT x86 setuid 0	60
EXPLOIT x86 stealth noop	32
EXPLOIT x86 setgid 0	30
EXPLOIT x86 NOPS	2

Top Ten Destination IPs	**# Total**
MY.NET.24.8	67
MY.NET.190.93	47
MY.NET.202.26	20
MY.NET.150.101	18
MY.NET.203.130	15
MY.NET.130.64	12
MY.NET.197.6	9
MY.NET.5.15	8
MY.NET.5.55	7
MY.NET.29.8	7

Table 3-17: x86 Top Talkers

MY.NET.24.8 seems to be the NEWS server. The servers 131.118.254.130, 128.8.5.30, 140.121.175.75 and 128.8.10.18 communicated with MY.NET.24.8 on destination port 119 (NNTP). All the source addresses mentioned belong to University of Maryland.

Hosts from network 131.118.254.0 and 128.8.5.30 showed up in NOOP alerts. These networks belong to University of Maryland. The alerts showed communication with MY.NET.24.8 (University internal NNTP server as shown in hosts Table earlier).

```
OrgName:   University of Maryland
NetRange:   131.118.0.0 - 131.118.255.255
CIDR:     131.118.0.0/16
NetName:    MINCNET NetHandle:  NET-131-118-0-0-1
NetType:    Direct Assignment

OrgName:    University of Maryland
Address:    Network Operations Center Bldg 224, Room 1301
City:       College Park StateProv:  MD PostalCode: 20742 Country:    US
 NetRange:   128.8.0.0 - 128.8.255.255
CIDR:       128.8.0.0/16
NetName:    UMDNET   NetHandle:  NET-128-8-0-0-1
```

Table 3-18: Who is database search results

Overall, there is good amount of noise in these alerts. The destination ports are associated with web servers, file sharing and other binary transfer operations. As mentioned in the Snort rule explanation at http://www.snort.org/snort-db/sid.html?sid=649 the chance for false positives is very high.

> 'Large binary transfers, certain web traffic, and even mail traffic can trigger this rule'.

Recommendations:

- It is important to reduce the number of alerts since there is a chance for real attacks getting overlooked because of the high volume of Exploit x86 alerts. Modify the Snort rule set to recognize known host networks.
- Set access control lists to block HTTP traffic to networks MY.NET.5.0 and MY.NET.29.8, if there is no need for these servers to serve outside networks.
- Apply web server patches on all the hosts serving HTTP from these networks.

3.4.3.7 Queso fingerprint

Classification: Noise (False Positive)

Queso fingerprint is used to guess the operating system type. Older queso signatures check the TCP headers to see if the two reserved bits in the TCP header are set in the initial SYN packet during TCP handshake. However, the proposed standard for Explicit Congestion Notification (ECN) may clash with Queso fingerprint detection as described by Toby Miller in his article http://www.securityfocus.com/infocus/1205.

> ECN is a standard proposed by the IETF that will cut down on network congestion and routers dropping packets. Currently, RFC 2481 states that in order to accomplish this task ECN will use four previously unused bits in both the IP header and the TCP Header. Queso sets the same reserved TCP bits that are being used by ECN.

There is also a newer version of snort signature to check for other features of Queso like high TTL value.

Source IP	# Total	Destination IP	# Total
66.117.30.14	63	MY.NET.24.23	80
210.253.206.180	32	MY.NET.24.21	78
213.186.35.9	27	MY.NET.24.22	58
216.95.201.34	23	MY.NET.6.40	57
81.57.90.18	22	MY.NET.6.47	51
212.202.170.228	22	MY.NET.211.26	43
209.47.197.17	21	MY.NET.224.134	32
209.47.197.18	21	MY.NET.217.54	25
66.140.25.157	19	MY.NET.24.44	24
216.95.201.37	17	MY.NET.237.118	23

Table 3-19: Queso Finger Print - Top Talkers

It is not clear if the rule used here is a new or older version.

Correlation: As can be seen from this discussion at http://archives.neohapsis.com/archives/postfix/2001-03/0583.html there seems to be some false positives for this alert.

Some of the internal hosts involved are the SMTP servers from the hosts table. There are also many related scans to these hosts during this time as can be seen from the scan logs.
It is easy to find SMTP banner information once an open TCP port like SMTP is found by establishing a telnet session to that particular host on the open TCP port. It is not very clear if these scans are regular scans or Queso scans. This alert creates more noise than any useful information.

Recommendations:

- Use modified signatures for specific fingerprints like Queso. Follow the ingress and egress filtering as described in section 3.7.

3.4.3.8 EXPLOIT NTPDX buffer overflow

Classification: False Positive

This alert indicates buffer overflow attempts against the NTP daemon. Network Time Protocol Daemon (ntpd) shipped with many systems is vulnerable to remote buffer overflow attack. Most of the times, ntpd is run with super user privileges, allowing remote users to gain REMOTE ROOT ACCESS to timeserver.

This attack appears to be a false positive. But on further analysis it is seen the host MY.NET.196.193 is also involved in other alerts including port 69 (TFTP) and port 31337 (Back Orifice).

As seen from the scan analysis the network MY.NET.196.0 seems to have some compromised hosts (MY.NET.196.193) scanning outside world for port 17300, a port associated with parasitic meta Trojan as described under scan log analysis in section 3.5.1.

NTP is stateless UDP based protocol, so all malicious queries can be spoofed. Some recent GCIA papers suggested limiting NTP to known NTP public servers. Though it helps to a large extent, since NTP uses UDP, it is easy to spoof a public time server address to exploit any NTP related vulnerabilities.

Recommendations:

- Patch the NTP servers with the latest updates.
- Move servers communicating directly with public timeservers into DMZ (Demilitarized Zone).
- Setup separate local timeservers for local area network to synchronize with DMZ timeservers.

3.4.3.9 IRC evil - running XDCC

Classification: Risk

The main purpose of XDCC is IRC (Internet Relay Chat) file sharing though it can be used to create bot (slave machines). The main function of XDCC is to automate listing of shared files on IRC channels for people to download.
A very good article written by Tonik Gin is available at http://www.russonline.net/tonikgin/EduHacking.html.

There are many straight hits in this case.
Table 3-20 below shows the top-ten internal and external IP addresses involved in the XDCC evil alerts.

Internal Address	External Address
MY.NET.150.205	216.152.65.144
MY.NET.207.78	217.17.33.10
MY.NET.241.246	193.163.220.3
MY.NET.86.33	157.156.254.222
MY.NET.80.209	160.94.151.137
MY.NET.249.250	63.98.19.244
MY.NET.132.24	38.192.23.234
MY.NET.112.199	193.216.236.142
MY.NET.198.221	216.32.207.207

Table 3-20: IRC evil running - Top Talkers

A link graph has been created to show the hosts involved in this attack and the direction of flow. But as found the link-graph leads to other alerts including red worm in link graph in Fig. 8.

Link Graph Figure 6:
As seen from the link graph it is very clear the hosts involved in this alert are compromised or on the verge of getting compromised. The linking arrows show the traffic flow direction. An internal host with no outgoing traffic may not necessarily mean an uncompromised host. It could be because there was no communication during this particular period or it is just a matter of time. The link graph is created for the purpose of understanding the attack and does not show all the addresses involved.

The analysis was started with address MY.NET.150.205. Though the IP addresses shown in the link graph are not directly involved in red worm alerts, the second link graph in Figure 8 shows the link between XDCC channel and red worm alerts. IP addresses MY.NET.105.204, 198.163.214.2 and 66.118.159.153 are common to two link graphs in Figure 7 and Figure 8 and are shown in a different color to represent their association to both link graphs.

Recommendations:

- All the hosts involved must be checked and cleaned for any backdoor vulnerability. Install any system-related patches.
- Enable ingress and egress filtering.

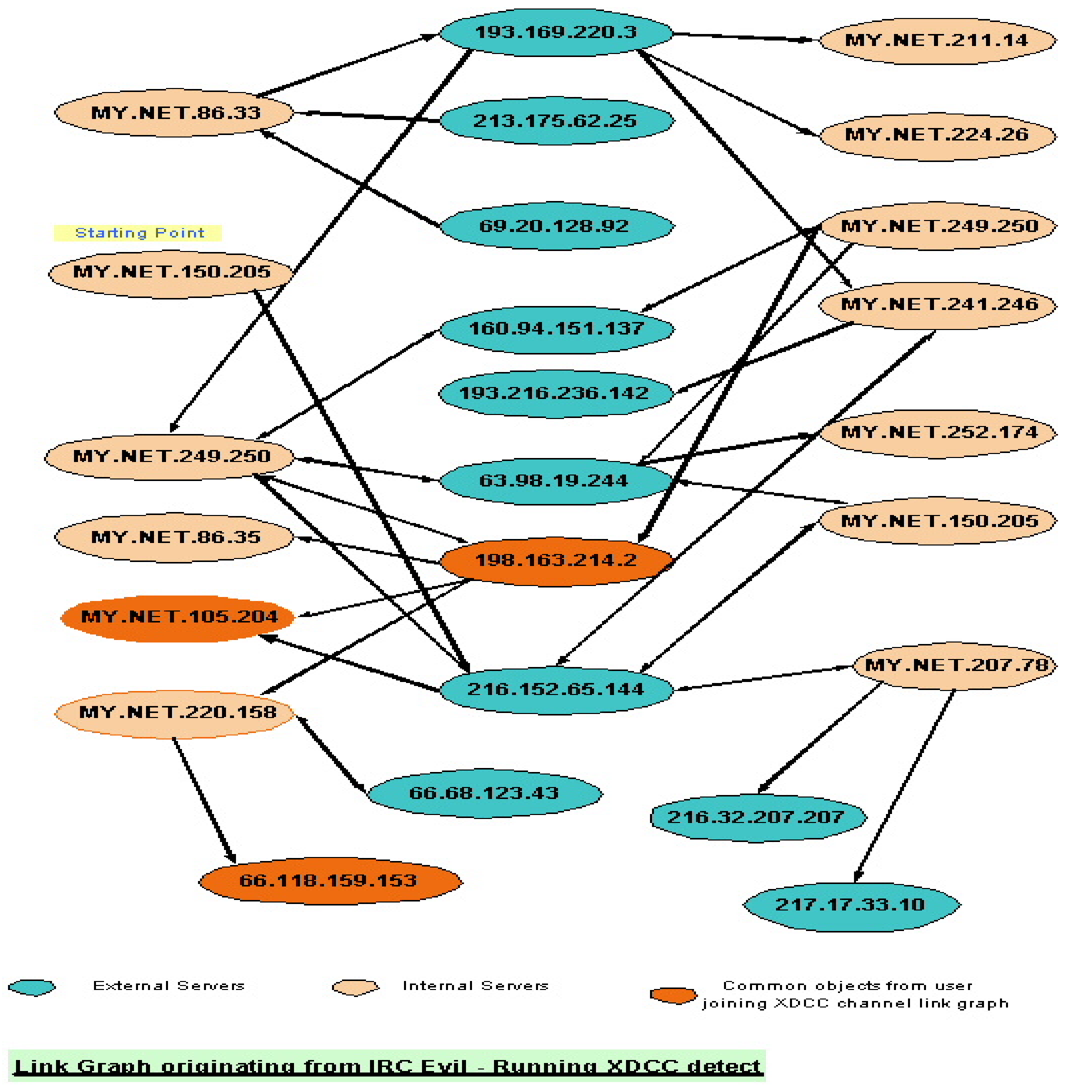


Figure 6: Link Graph originating from IRC evil running XDCC

3.4.3.10 [UMBC NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot

Traffic Flow: Incoming – 0 Outgoing - 258
Unique hosts: Incoming – 0 Unique Outgoing - 5
All source ports are 6667.

Warez is a term used to describe a cracked version of commercial software or applications. Warez channels are used to distribute copyrighted files like cracks, software, movies etc. Table 3-21 below shows the top source and destination hosts from warez channel detect.

Source IP	# Total	Destination IP	# Total
209.126.191.127	6667	MY.NET.82.241	1090
66.118.185.29	6667	MY.NET.240.42	4533
66.163.242.103	6667	MY.NET.201.6	2870

66.93.61.143	6667	MY.NET.201.26	2018
207.54.31.254	6667	MY.NET.224.14	4404

Table 3-21: IRC User joining Warez - Top Talkers

Also alerts with source port=6667 are 258 and destination port=6667 are 5413. MY.NET.201.26 is also involved in twenty-six IIS Unicode attacks with 5 destinations - 211.233.29.57, 211.233.29.54, 211.233.29.3, 211.233.78.77 and 211.233.79.230. All addresses belong to Korean network. IIS Unicode alerts can result from regular web surfing if the web content contains characters matching the signature rule contents. There is a high chance for false positives with Unicode alerts. But repeated alerts to the same destination should not be ignored. MY.NET.224.14 is involved in TCP scans.

Recommendations:
Check the machines MY.NET.201.26 and MY.NET.224.14 for backdoor programs.

3.4.3.11 [UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot

XDCC can be used to make bot out of compromised systems to serve Warez software to the users of IRC channels.
These alerts have 6666, 6667 and 6668 in destination ports. It is easy to see the relationship between the sources and destinations from the table. But as happened with IRC – evil XDCC a simple check revealed a link to another alert. Table 3-22 below shows the top source and destination hosts from XDCC channel.

Source IP	# Total	Destination IP	# Total
209.6.189.30	6667	MY.NET.226.102	4992
208.178.231.190	6667	MY.NET.224.14	1838
217.67.227.5	6667	MY.NET.194.201	3894
198.163.214.2	6661	MY.NET.220.158	1514

Table 3-22: User joining XDCC channel - Top Talkers

MY.NET.226.102, 209.6.189.30 and 198.163.227.5 are involved in other alerts. Since there are good number of alerts associated with 198.163.214.2 a quick check was done against 198.163.214.5 and it revealed an interesting situation. The IP addresses shown in the link graph in Figure 7 represent IRC and red worm related traffic. IP addresses MY.NET.105.204, 198.163.214.2 and 66.118.159.153 are common to both link graphs in Figure 6 and Figure 7 and are shown in a different color to represent their association to both link graphs. For example there is two-way red worm traffic between IP addresses MY.NET.233.286 and 66.118.159.153.

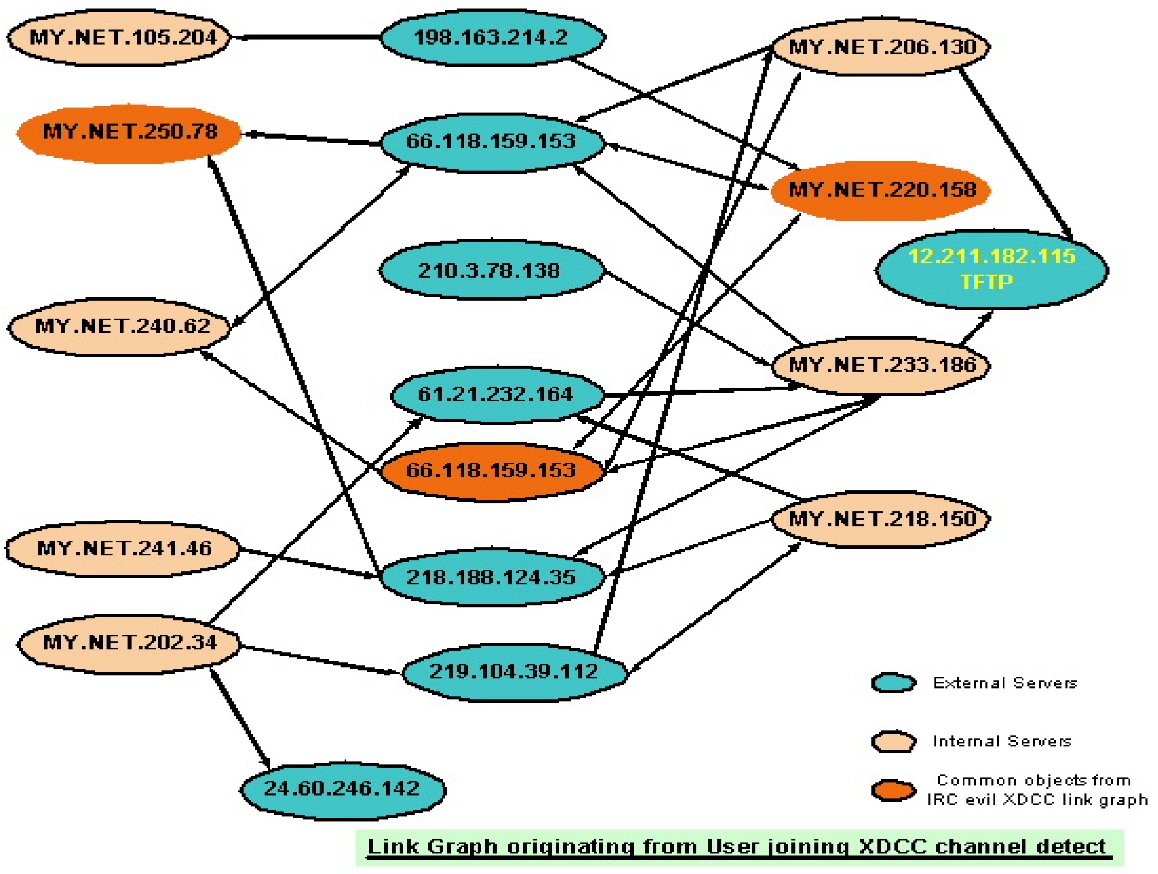


Figure 7: Link Graph - IRC XDCC and Red Worm

Recommendations:

- Generic detection rules generated a large number of false positives showing traffic from peer-to-peer network as red worm traffic. The Snort rules need fine-tuning.
- The networks involved need thorough check. Refer to the recommended audit methods mentioned under section 3-4.

3.4.3.12 Possible Trojan server activity

Since this alert appears to be very generic, a list of top source and destination ports involved has been created. The top port involved is 27374, a port associated with a long list of Trojans and worms. All ports appeared excluding port 0, 80 and 999 are ephemeral ports and as such can be used by any application. A quick search against alert and scan logs has shown communication between two IP addresses 216.170.7.20 and MY.NET.201.182. Table 3-23 below shows the top source and destination hosts with corresponding ports from possible Trojan server activity.

Source Ports	Destination Ports	Source Hosts	Destination Hosts
27374	27374	MY.NET.202.54	213.8.239.155
1826	1826	MY.NET.235.42	66.32.72.20
1263	1263	MY.NET.202.226	24.125.13.198

2495	1214	MY.NET.194.35	62.195.161.26
1214	2495	MY.NET.218.46	193.251.75.254
80	80	MY.NET.24.34	66.56.197.97
6884	0	MY.NET.24.44	4.35.71.83
3141	6884	MY.NET.5.20	65.41.74.246
3833	999	MY.NET.211.110	68.61.209.109
1258	6346	MY.NET.194.13	67.2.145.186

Table 3-23: Possible Trojan - Top Talkers

Host MY.NET.201.182 is involved in multiple alerts to hosts 80.7.81.92 and 81.53.178.107 and scans from 219.195.12.10 and 61.242.166.150. All the scans to outside world used source port 6042. MY.NET.201.182 is also involved in extern UDP alerts for TFTP.

Obviously there are some hits from this rule but this very generic rule set makes it difficult to sort through this huge alert list. There is a good chance for having many false positives.

Recommendations:
- It would make things difficult if a rule set is based on just port numbers while dealing with ports like 27374. Content specific rule sets should be considered to replace this more generic rule.
- Keeping the current rule along with other more specific rules sets would certainly help create a traffic pattern.

3.4.3.13 All bot alerts

A bot is a script or program that is designed to perform automated actions on behalf of the owner (the attacker), usually without owner's direct intervention. Bot can be programmed as part of a normal chat client such as IRC. Bot serve different purposes. Bot can be used to monitor and transfer end user's actions like capturing keystrokes, participating in denial of service attacks, looking for other vulnerable systems etc.
A generic search for the word 'bot' in alerts showed the following source and destination addresses in table 3-24.

Source IP	**# Total**	**Destination IP**	**# Total**
MY.NET.97.79	44	216.152.64.155	114
MY.NET.97.85	34	MY.NET.224.14	2
MY.NET.97.93	24	MY.NET.194.201	1
MY.NET.97.97	5	MY.NET.82.241	1
MY.NET.97.242	4	MY.NET.220.158	1
MY.NET.97.72	3	MY.NET.240.42	1
198.163.214.2	1	MY.NET.226.102	1
209.126.191.127	1	MY.NET.201.6	1
208.178.231.190	1	MY.NET.201.26	1

Table 3-24: Consolidated bot alerts - Top Talkers

The destination is 216.152.64.155. All the ports used by this address are used extensively by IRC related activity. All IP addresses are related to sdbot floodnet and IRC user/kill detects. MY.NET.97.0 network is vulnerable and has some compromised machines.

Source IP addresses communicating with 216.152.64.155	**# Total**	**Ports used by 216.152.64.155**
MY.NET.97.79	44	6666
MY.NET.97.85	34	6663
MY.NET.97.93	24	6664
MY.NET.97.97	5	6660
MY.NET.97.242	4	6661
MY.NET.97.72	3	6665

Table 3-25: Internal Sources to 216.152.64.155

216.152.64.155 (No scans or OOS) NetRange: 216.152.64.0 - 216.152.79.255 CIDR: 216.152.64.0/20 NetName: WEBMASTER-BLK-1 NetHandle: NET-216-152-64-0-1 NetType: Direct Allocation OrgName: WebMaster, Incorporated OrgID: WBMR Address: City: Santa Clara StateProv: CA PostalCode: 95050 Country: US

Table 3-26: Who is database search results

Recommendations:

- Check the machines in table 3.25 for any vulnerabilities and compromises.

3.4.3.14 NIMDA – Attempt to execute cmd from campus host

Traffic Flow: Incoming – 0 Outgoing – 75
Unique hosts: Incoming – 0 Destination Port - 80

The name of the worm Nimda came from the reversed spelling of 'admin'. This worm attempts to infect unpatched Microsoft IIS web servers by using directory traversal vulnerability. The worm also attempts to use IIS server that had previously been compromised by Code Red II.
Network MY.NET.97.0 and MY.NET.98.0 are involved. MY.NET.97.0 appears one more time. Table 3-27 below shows the top internal sources from nimda alerts.

Internal Sources	**# Total**
MY.NET.97.67	25
MY.NET.97.19	22
MY.NET.97.44	11
MY.NET.97.18	7
MY.NET.98.146	7

MY.NET.97.122	2
MY.NET.97.165	1

Table 3-27: Nimda - Top Talkers

The alert and scan logs correlate with the sources shown in the above table participating in 'execute cmd' attacks as well as random scans of outside networks from internal hosts. Some of the addresses also appear in the top-talkers list shown in scans analysis in later sections.
There should not be any scans from internal sources. These machines are compromised.

Recommendations:

- Update all patches.
- Move IIS root from default C drive to some other drive.
- Use tools to find back doors and Trojans.

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

3.4.3.15 Back Orifice

Classification: False Positive

Back Orifice is a remote access Trojan. The alert signature was set to capture traffic going to port 31337. Three outside IP addresses were found talking to 3 internal IP addresses. The scan files showed two IP addresses scanning the hosts during this time.
Table 3-28 below shows the top source and destination hosts and corresponding ports from Back Orifice alerts.

Source IP	Source Port	Destination IP	Destination Port
211.177.191.184	2352	MY.NET.196.29	31337
211.221.85.195	2949	MY.NET.196.29	31337
207.207.20.129	3862	MY.NET.196.29	31337

Table 3-28: Back Orifice - Top Talkers

MY.NET.196.29 has other connections from 4.62.30.53, 211.49.82.138 and 218.159.24.157 (external udp to internal tftp), (EXPLOIT NTPDX buffer overflow) to 218.144.69.214, 211.117.118.53, 63.97.39.185, 211.47.87.252, 217.37.119.133. There are no scans or alerts with MY.NET.196.29 as the source address. If this machine is compromised more alerts would have shown up with this address as the source. The scan logs show UDP scans from 211.177.191.184 and 207.207.20.129 to MY.NET.196.29. The other alert from 211.221.85.195 is not associated with any corresponding alerts from scan and oos logs. That could be because of a regular connection that happened to have 31337 as the destination port. This alert is classified as false positive.

Correlation: Tod Beardsley gave similar analysis in his white paper.
http://www.giac.org/practical/Tod_Beardsley_GCIA.doc

Recommendations:

- Update to latest Snort signatures. The new signatures allow for Back Orifice related content checking in the payload helping reduce the false positives.

3.5 Scans Analysis

Total Scans: 737583
Total Unique Internal Hosts as source: 403
Total Unique External Hosts as source: 38999

3.5.1 Top Talkers in Scans

Table 3-29 below lists internal and external IP addresses participating in scans. The addresses shown in red color appeared in dsheild database as offenders. Ideally there should not be any scans originating from internal network. Some scanning could result from internal scanning by security groups.

Top Ten Internal Sources		Top Ten External Sources	
Internal IPs	**# Total**	**External Ips**	**# Total**
MY.NET.196.193	238876	162.39.73.134	3810
MY.NET.87.50	38649	80.11.48.186	3561
MY.NET.197.30	22363	69.10.15.110	3352
MY.NET.225.154	9958	80.13.0.75	3242
MY.NET.97.19	8003	209.120.166.71	2503
MY.NET.250.162	7951	207.194.62.117	2471
MY.NET.97.18	7898	195.199.70.26	2363
MY.NET.217.62	7637	213.84.138.106	2313
MY.NET.97.67	5891	61.242.166.150	2290
MY.NET.220.62	5828	149.68.10.6	2144

Table 3-29: Top 10 Internal and External IPs - From Scan Logs

Table 3-30 below shows all the different scans that occurred during the five-day analysis period.

Scan Type	# Total
SYN	369547
NULL	362625
FIN	2668
NOACK	1761
INVALIDACK	939
SYNFIN	39

Table 3-30: All Scans from Scan Logs

Table 3-31 below lists all the internal IP addresses associated with scanning activity. A thorough check has to be done on these machines for backdoor Trojans and any other activity related to security compromise.

Top Source IPs		Top Destination IPs	
MY.NET.196.193	MY.NET.97.98	MY.NET.130.85	MY.NET.85.163
MY.NET.203.198	MY.NET.211.210	MY.NET.85.62	MY.NET.85.235
MY.NET.97.18	MY.NET.227.6	MY.NET.85.107	MY.NET.85.75

MY.NET.234.190 MY.NET.233.2 MY.NET.97.52 MY.NET.240.190 MY.NET.84.151 MY.NET.242.66	MY.NET.97.44 MY.NET.222.26 MY.NET.250.226 MY.NET.205.50 MY.NET.221.202	MY.NET.85.146 MY.NET.85.150 MY.NET.85.130 MY.NET.85.64 MY.NET.85.27 MY.NET.85.74	MY.NET.85.58 MY.NET.85.15 MY.NET.85.17 MY.NET.85.77 MY.NET.85.2

Table 3-31: Top Talkers from Internal Network

The top three scanned ports are shown below. The earlier sections showed the analysis for these ports. The top ranked internal source host from scan logs MY.NET.196.193 is found running SYN scans looking for open port TCP/17300.

Source Ports	Protocol	# Total	Destination Ports	Protocol	# Total
27022	UDP	38650	17300	TCP	248588
7674	UDP	13529	80	TCP	52971
22321	UDP	11496	137	UDP	38314

Table 3-32: Top 3 TCP/UDP Port Scans for both Source & Destination Ports from Scan Logs

This port is associated with a parasitic meta Trojan named Milkit by Lurhq. More information on milkit is at http://www.lurhq.com/sig-milkit.html. MY.NET.196.193 appears as a compromised host, acting as a slave to remote host commands.

There were many SMB alerts from network 216.176.0.0 to MY.NET.196.193 along with other IP addresses.

05/16-07:08:24.376086 [**] SMB Name Wildcard [**] 216.176.80.86:137 -> MY.NET.196.193:137

There were scan alerts from host MY.NET.196.197 scanning network 216.176.0.0

*May 16 07:08:22 MY.NET.196.193:4070 -> 216.176.80.123:17300 SYN ******S**

Though it appears that the scan from MY.NET started before the SMB alert came in from MY.NET.196.193, overall scanning from MY.NET.196.193 has been going on for a long time before and after the traffic to port 137 came in. It is not clear if the SMB alerts from outside network are a way to obfuscate the command but the SMB alerts came from the same networks as the scans from the compromised hosts are going to.

Similarly host MY.NET.202.202 is using source port 1470/UDP to scan open UDP ports. Host MY.NET.222.166 is involved in many alerts and scans for port 80 including CGI null byte attacks.

Recommendations:
- Check the machines MY.NET196.197 and MY.NET.202.202 for backdoor Trojans.
- Implement other recommendations related to ingress and egress filtering as described in the executive summary.

3.6 OOS Analysis

Total OOS events during the five day analysis are 4847.

OOS packets are out of specification packets. These logs represent packets not conforming to RFC standards. The major offenders appeared in all log files. The top-talkers specifically appeared under queso finger print alerts, and scans with reserved TCP bits set and OOS logs.

These are some of the TCP flag combinations that do not conform to TCP RFC standards.

SYN and FIN set, SYN and RST set, SYN/FIN/PSH set, SYN/FIN/RST set, SYN/FIN/RST/PSH set, etc. Since a SYN packet is used to initiate a TCP connection, it should never have the FIN or RST flags set in conjunction. It is always a malicious attempt at getting past the firewall or router access control lists.

FIN (without ACK)
An ACK bit should always accompany a FIN packet, since the only reason to send an ACK/FIN packet is to tear down an existing connection.

"NULL" packet
A "NULL" packet is a packet with no TCP flags set.

Reserved flags set. As discussed in queso finger print analysis this is a recent RFC for explicit congestion notification and may cause issues if used against non-ECN compatible hosts.

The reason for an illegal TCP flag combination could be the following reasons.
1) The packet got corrupted during creation (TCP stack problem) or during network transmission.
2) The packet is following RFC2481 ECN standards.
3) The packet is crafted for malicious attacks.

Table 3-33 below shows TCP flag combinations along with the top-talkers from OOS logs.

TCP Flags	# Total	Source IP	# Total	Destination IP	# Total
12****S*	4409	66.117.30.14	492	MY.NET.224.134	295
****P***	314	81.57.90.18	174	MY.NET.6.47	241
********	40	210.253.206.180	159	MY.NET.6.40	236
12***R**	21	209.123.49.137	136	MY.NET.211.26	231
*2U*PRSF	9	212.202.170.228	118	MY.NET.24.22	230
**UAPRSF	3	213.186.35.9	89	MY.NET.24.21	227
12*APR**	3	212.186.78.246	88	MY.NET.24.23	215
*2U**RSF	3	209.47.197.17	84	MY.NET.237.118	154
*2****SF	2	196.26.86.133	73	MY.NET.233.78	141

12*****F	2	213.197.11.147	72	MY.NET.24.44	128

Table 3-33: All Flags, Source & Destination IPs from OOS Logs

3.6.1 Out of Spec Details:

The following is an analysis of source 81.57.90.18, which appeared in all log files at the same time. As described under queso finger print analysis, the reserved bits are set here. And this could be a genuine implementation according to RFC2481 (Explicit Congestion Notification), or a crafted packet or packet corruption on the wire.

```
From Alert Logs:
05/16-03:40:33.402906  [**] Queso fingerprint [**] 81.57.90.18:34741 ->
MY.NET.218.154:6346
05/16-05:12:19.852123 81.57.90.18:34348 -> MY.NET.224.122:6346

From Scan Logs:
May 16 03:40:33 81.57.90.18:34741 -> MY.NET.218.154:6346 SYN 12****S*
RESERVEDBITS
May 16 05:17:02 81.57.90.18:38644 -> MY.NET.85.218.154:6346 SYN 12****S*
RESERVEDBITS
From OOS Logs:
-------------------------------------------------------------------------------------------
05/16-03:40:33.402910 81.57.90.18:34741 -> MY.NET.218.154:6346
TCP TTL:47 TOS:0x0 ID:50463 IpLen:20 DgmLen:60 DF
12****S* Seq: 0xD6ACF9A1  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 13778110 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Table 3-34: Log details for 81.57.90.18

As noted in queso finger print analysis earlier, the top destination port is 25 (SMTP). The top destination ports also included file-sharing ports like 4662 used by file sharing applications like edonkey. In the above logs the destination port 6346, associated with Gnutella, indicates a possible Gnutella peer on the other side trying to contact Gnutella peer inside. Tod Beardsley in his GCIA paper shows something similar. http://www.giac.org/practical/Tod_Beardsley_GCIA.doc

In this case, there are no log entries to indicate packet retransmission. In case of a corrupted packet the host resends the packet until the packet is acknowledged or there is a time out. However in case of a valid packet the logs would not capture a re-sent packet. But as seen from the other entries it is clear that there has been a stealth scan from source 81.57.90.18 to other hosts on port 6346. That indicates that in fact this is a reconnaissance attempt to gather information on open file sharing ports for the kinds of Gnutella, edonkey etc. Most probably these peer hosts are sitting behind a RFC2481 enabled router. This rules out packet corruption on the wire.

On the other hand the following analysis indicates crafted packets. Table 3-40 below shows a probable packet-crafting scenario from OOS logs. Source address 213.186.35.9 is trying gather information on MY.NET.97.48. The closeness of the header values, marked in bold letters, across the scan is an indication of a basic packet crafting tool in action.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/16-20:04:37.177029 213.186.35.9:58833 -> MY.NET.97.48:81
TCP TTL:49 TOS:0x0 ID:3401 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x88A5450A  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 5392451 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ =+=+=+=+
05/16-20:04:37.177070 213.186.35.9:58836 -> MY.NET.97.48:1080
TCP TTL:49 TOS:0x0 ID:28871 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x88F03884  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 5392451 0 NOP WS: 0
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/16-20:04:37.177228 213.186.35.9:58838 -> MY.NET.97.48:23
TCP TTL:49 TOS:0x0 ID:50965 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x88418F56  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 5392451 0 NOP WS: 0
```

Table 3-35: Packet crafting example from OOS logs

Table 3-36 below shows excerpts from oos logs for source address MY.NET.12.4 and MY.NET.12.2. Address MY.NET.12.4 is serving mail clients over IMAP (143) and IMAP over SSL (993). Address MY.NET.12.4 is serving SMTP. The reserved flags are set.

Time	Src IP:Src Port	Dst IP:Dst Port	TCP Flags
2003-05-15 08:37:53	**MY.NET.12.4:993**	68.55.193.137:28926	**12***R****
2003-05-16 10:34:50	MY.NET.12.4:993	172.156.181.171:49235	12***R**
2003-05-16 17:20:09	MY.NET.12.4:**143**	205.244.232.133:80	12***R**
2003-05-16 22:20:41	MY.NET.12.4:143	216.5.176.162 :80	12***R**
2003-05-18 01:49:50	MY.NET.12.4:143	68.32.61.11:1042	12***R**
2003-05-18 11:36:56	MY.NET.12.4:143	138.88.147.50:51544	12***R**
2003-05-17 09:38:24	MY.NET.12.4:993	68.55.193.137:29202	12***R**
2003-05-17 22:05:47	MY.NET.12.4:143	68.55.205.180:4717	12***R**
2003-05-19 09:09:43	MY.NET.12.4:993	10.0.1.8:49315	12***R**
2003-05-19 12:52:27	MY.NET.12.4:993	68.32.128.55:49153	12***R**
2003-05-15 00:19:12	**MY.NET.12.2:25**	192.168.1.100:1024	**12***R****
2003-05-18 14:08:53	MY.NET.12.2:25	81.78.117.33:3641	12***R**
2003-05-17 18:55:31	MY.NET.12.2:25	172.166.234.65:1661	12***R**

Table 3-36: RST flag set witn Reserved bits

As mentioned at http://www.zvon.org/tmRFC/RFC3360/Output/chapter3.html

> When a TCP host negotiating ECN sends an ECN-setup SYN packet, an old TCP implementation is expected to ignore those flags in the Reserved field, and to send a plain SYN-ACK packet in response. However, there

are some broken firewalls and load-balancers on the Internet that instead respond to an ECN-setup SYN packet with a reset.

The destination IP addresses belong to dialup and cable modem users. These machines might be sitting behind ECN enabled routers. A few NULL scan alerts were found from MY.NET.12.4 and MY.NET.12.2 but it is probably a bad packet from TCP stack. Since there are not many entries in the logs no action is needed.

3.7 Recommendations:

Network Perimeter:

- There is no need to allow incoming traffic for ports other than 25, 53, 80, 119 and 443 and any other specific public servers. Block incoming traffic for other ports. Setup incoming traffic only for TCP established connections, connections initiated from inside.
- Allow incoming UDP only if absolutely needed.
- Setup access-control lists for incoming 53 (DNS) traffic to allow traffic from site specific DNS servers only.

Audit:

- Perform thorough check of suspected machines for Trojans and backdoors. Audit open ports in the internal network using a tool like nmap to scan and find open ports and other unauthorized services. Use tools like nessus to find service vulnerabilities before moving servers into production.

Others:

- Setup anti-virus at SMTP gateways and on client machines. Sanitize banner information on public servers.

Move the public servers to DMZ and setup access-control lists to allow traffic from DMZ to outside world and intranet.

- Fine-tune Snort rules. Some of the rules are very generic resulting in high volume of false positives.

3.8 References:

The following resources were referenced directly in this section, or were referred to for information and not otherwise noted:

MySql http://www.mysql.com
Search Engine http://www.google.com
Who is database http://www.whois.sc http://www.arin.net
http://isc.incidents.org
http://www.ietf.org
http://www.dshield.org
http://www.sans.org/resources
http://dast.nlanr.net/Projects/Beacon
Unix Mail Archives
    http://www.health.ufl.edu/mail-archives/unix-sec
    http://archives.neohapsis.com/archives/postfix/2001-03/
http://www.securityfocus.com

http://xforce.iss.net/
http://archives.neohapsis.com/archives/postfix/2001-03/0583.html
http://www.securiteam.com/unixfocus/NTPD_vulnerable_to_a_remotely_exploitable_buffer_overflow__readvar_.html
http://www.linuxsecurity.com/advisories/netbsd_advisory-1255.html
http://www.whitehats.com/info/IDS492
http://www.cert.org/advisories
http://www.lurhq.com/sig-milkit.html
http://www.seifried.org/security/ports/
XDCC http://www.russonline.net/tonikgin/EduHacking.html
Tiny Fragments
http://www.cisco.com/en/US/tech/tk827/tk369/technologies_white_paper09186a00800d6979.shtml
http://www.giac.org/practical/Mark_Embrich_GCIA.htm
Stealth noop http://www.snort.org/snort-db/sid.html?sid=651
X86 NOOP http://www.snort.org/snort-db/sid.html?sid=648
Setgid http://www.snort.org/snort-db/sid.html?sid=649
Setuid http://www.snort.org/snort-db/sid.html?sid=650

Nmap scanning http://www.insecure.org/nmap/nmap_doc.html
RPC
http://probing.csx.cam.ac.uk/about/sunrpc.html
IRC and XDCC
http://www.russonline.net/tonikgin/EduHacking.html
Specific example of ECN
http://www.zvon.org/tmRFC/RFC3360/Output/chapter3.html
http://www.giac.org/practical/GCIA/Michael_Cloppert_GCIA.pdf
http://nwn.bioware.com/support/techfaq.html#03

4. Appendix A

4.1 Tools and Scripts used in "Analyze This!"

```
#!/usr/bin/perl
##############################################
##      Program:        parsealerts.pl
##      Author:         Sai Prasad Kesavamatham
##      To parse and import alerts logs
###############################################

###### Change variables here ######
$Year = 2003;
$WorkingDirectory = "/gcia/analyze/alerts";
$MySqlDirectory = "/mysql/bin";
###### End of Variables ###########
```

```
use File::Copy;
opendir (ALERTDIR, "$WorkingDirectory");
chomp(@Files = readdir(ALERTDIR));
close (ALERTDIR);

shift(@Files);
shift(@Files);

 foreach $File(@Files){
  open (ALERTS, "$WorkingDirectory/$File");
  chomp(@AlertsArray = <ALERTS>);
  close (ALERTS);
  $LogFile = "$File.parsed.txt";

  open (LOG, ">$WorkingDirectory/$LogFile");

   while (@AlertsArray){
    $AlertLine = shift(@AlertsArray);
    ($TimeStamp, $Alert, $Address) = split (/\[\*\*\]/, $AlertLine);
      if ($Alert =~ /spp_portscan/){
            goto NextLine;
      }
        ($Month, $Day, $Time) = split (/\/|\-/, $TimeStamp);
        ($Hour, $Minute, $Second, $MilliSecond) = split (/\:|\./, $Time);
        $Alert =~ s/^\s+//;     #cut leading space off alert
        $Alert =~ s/\s+$//;     #cut trailing space off alert
        $Alert =~ s/\s+$//;     #cut trailing space off alert
          $MilliSecond =~ s/\s+$//;
        ($SourceAddress, $DestinationAddress) = split (/\->/, $Address);
        ($SourceIP, $SourcePort) = split (/\:/, $SourceAddress);
          $SourcePort =~ s/\s+$//;
          $SourceIP =~ s/^\s+//;
        ($DestinationIP, $DestinationPort) = split (/\:/, $DestinationAddress);
          $DestinationIP =~ s/^\s+//;
if ($Alert ne ""){
print LOG <<EOM;
'null','$Year$Month$Day$Hour$Minute$Second','$MilliSecond','$Alert','$SourceIP
','$SourcePort','$DestinationIP','$DestinationPort'
EOM
}
NextLine:
    }
  close (LOG);
copy "$WorkingDirectory/$LogFile" => "$WorkingDirectory/alerts.txt";
system ("d:/mysql/bin/mysqlimport -L logs -u gcia --fields-terminated-by=, --fields-
enclosed-by=' $WorkingDirectory/alerts.txt");
unlink ("$WorkingDirectory/alerts.txt");
 }
```

Table 4-1: parsealerts.pl

```
#!/usr/bin/perl
##############################################
##      Program:        parsescans.pl
##      Author:         Sai Prasad Kesavamatham
##      To parse and import scan logs
##############################################

###### Change variables here ######
$Year = 2003;
$WorkingDirectory = "/gcia/analyze/scans";
$MySqlDirectory = "/mysql/bin";
$Alert = "Scan";
$MYHOME = '130.85';
my %Months = ('Jan'=>'01','Feb'=>'02', 'Mar'=>'03','Apr'=>'04','May'=>'05',
   'Jun'=>'06','Jul'=>'07','Aug'=>'08','Sep'=>'09','Oct'=>10,'Nov'=>11,'Dec'=>12);
###### End of Vairables ########
use File::Copy;
opendir (SCANDIR, "$WorkingDirectory");
chomp(@Files = readdir(SCANDIR));
close (SCANDIR);
shift(@Files);
shift(@Files);

 foreach $File(@Files){
  open (ALERTS, "$WorkingDirectory/$File");
  chomp(@AlertsArray = <ALERTS>);
  close (ALERTS);
  $LogFile = "$File.parsed.txt";

  open (LOG, ">$WorkingDirectory/$LogFile");

   while (@AlertsArray){
    $AlertLine = shift(@AlertsArray);
    @Fields = split(/\s+/,$AlertLine);
    $Month = @Months{@Fields[0]};
    $Day = @Fields[1];
    ($Hour,$Minute,$Second) = split(/\:/,@Fields[2]);
    ($SourceIP,$SourcePort) = split(/\:/,@Fields[3]);
    ($DestinationIP,$DestinationPort) = split(/\:/,@Fields[5]);
    if (@Fields[6] eq "UDP"){
      $Protocol = "UDP";
```

```
     $Flag= "NULL";
   }
   elsif (@Fields[6] =~ /SYN/ || @Fields[6] =~ /FIN/ || @Fields[6] =~ /ACK/){
     $Protocol = "TCP";
     $Flag = "@Fields[6]";
   }
   elsif (@Fields[6] eq "NULL"){ }

   if ($SourceIP =~ /$MYHOME/){
     ($Oct1,$Oct2,$Oct3,$Oct4) = split(/\./,$SourceIP);
     $SourceIP = "MY.NET.$Oct3.$Oct4";
   }
   if ($DestinationIP =~ /$MYHOME/){
     ($Oct1,$Oct2,$Oct3,$Oct4) = split(/\./,$DestinationIP);
     $DestinationIP = "MY.NET.$Oct3.$Oct4";
   }

############

print LOG<<EOM;
'NULL','$Year$Month$Day$Hour$Minute$Second','$Alert','$SourceIP','$SourceP
ort','$DestinationIP','$DestinationPort','$Protocol','$Flag'
EOM
NextLine:
   }
  close (LOG);
copy "$WorkingDirectory/$LogFile" => "$WorkingDirectory/scans.txt";
# system ("d:/mysql/bin/mysqlimport -L logs -u gcia --fields-terminated-by=','
d:/mysql/bin/alerts.txt");
system ("d:/mysql/bin/mysqlimport -L logs -u gcia --fields-terminated-by=, --fields-
enclosed-by=' $WorkingDirectory/scans.txt");
unlink ("$WorkingDirectory/scans.txt");

 }
```

Table 4-2: parsescans.pl

```
#!/usr/bin/perl
############################################
##      Program:        parseoos.pl
##      Author:         Sai Prasad Kesavamatham
##      To parse and import oos logs
#############################################

###### Change variables here ######
$Year = 2003;
$WorkingDirectory = "/gcia/analyze/oos";
```

```
$MySqlDirectory = "/mysql/bin";
###### End of Variables ###########
use File::Copy;
# Read oos directory for files
opendir (OOSDIR, "$WorkingDirectory");
chomp(@Files = readdir(OOSDIR));
close (OOSDIR);
# Take out the dots
shift(@Files);
shift(@Files);

# Read each oos file and eliminate the lines with Hex etc and get the first line in
the entry
foreach $File(@Files){
  open (OOS, "$WorkingDirectory/$File");
  chomp(@OosArray = <OOS>);
  close (OOS);
  $LogFile = "$File.parsed.txt";
  open (LOG, ">$WorkingDirectory/$LogFile");
  foreach $OosLine(@OosArray){
    if ($OosLine =~ /\=\+/ || $OosLine eq "" || $OosLine =~ /\w\w\s\w\w\s/){
   #   goto NextLine;
    }
    elsif ($OosLine =~ /->/) {
      @Fields = split(/\s+/, $OosLine);
      ($Month, $Day, $Time) = split (/\/|\-/, @Fields[0]);
      ($Hour, $Minute, $Second, $MilliSecond) = split (/\:|\./, $Time);
      ($SourceIP, $SourcePort) = split (/\:/, @Fields[1]);
      ($DestinationIP, $DestinationPort) = split (/\:/, @Fields[3]);
    }
    elsif ($OosLine =~ /Seq:/){
      @Options = split(/\s+/, $OosLine);
      $Flags = @Options[0];
      $SeqNo = @Options[2];
      $AckNo = @Options[4];
      $Win = @Options[6];

# Get the same format as a parsed Alert file for database import
print LOG <<EOM;
'null','$Year$Month$Day$Hour$Minute$Second','$MilliSecond','OOS','$SourceIP',
'$SourcePort','$DestinationIP','$DestinationPort','$Flags','$SeqNo','$AckNo','$Win
'
EOM
    }

  NextLine:
  }
 close (LOG);
copy "$WorkingDirectory/$LogFile" => "$WorkingDirectory/oos.txt";
system ("d:/mysql/bin/mysqlimport -L logs -u gcia --fields-terminated-by=, --fields-
enclosed-by=' $WorkingDirectory/oos.txt");
unlink ("$WorkingDirectory/oos.txt");
}
```

Table 4-3: parseoos.pl

```
#!/usr/bin/perl
##############################################
##      Program:        common.pl
##      Author:         Sai Prasad Kesavamatham
## Use this script, if the database does not support intersect and
## join/union sql combination is a problem
## To find common IPs between two similar fileds across two tables
## However the same can be easily achieved using a combination of
## temporary tables as shown in the next table describing sql.
## Syntax: perl common.pl <table1> <table2> <field1> <field1.
## Ex. Perl common.pl alerts scans srcip srcip
## No error catching done.
###############################################
#!/usr/bin/perl
use DBI;
###### Change variables here ######
$user = 'userid';
$passwd = 'passwd';
$host = 'sqlserver1';
$Database = 'logs';
##### End of variables
$Table1 = $ARGV[0];
$Table2 = $ARGV[1];
$Field1 = $ARGV[2];
$Field2 = $ARGV[3];
# Connect to database
my $dbh = DBI->connect("DBI:mysql:$Database:$host","$user","$passwd",
    { RaiseError => 1, AutoCommit => 1})
            or die "Error: $DBI::errstr\n";

$SqlRef = &Sql("select $Field1,count(*) as val from $Table1 group by $Field1
order by val desc");

 foreach $href(@{$SqlRef}){
 # print "$href->{'$Field1'}\n";
   $Value = $href->{$Field1};
 push (@IPArray1,$Value);
 }
$SqlRef = &Sql("select $Field2,count(*) as val from $Table2 group by $Field2
order by val desc");

 foreach $href(@{$SqlRef}){
#  print "$href->{'srcip'}\n";
 push (@IPArray2,$href->{$Field2});
 }
$Counter = 0;
 foreach $T1(@IPArray1){
  foreach $T2(@IPArray2){
   if ($T1 eq $T2){
    print "$T1\n";
$Counter++;
   goto NextElement;
   }
  }
 NextElement:
 }

sub Sql {
    my @sql = @_;
    my @rows;
    my $sth = $dbh->prepare(@sql);
    my $rv = $sth->execute();
    if ($sql[0] =~ /^INSERT.*/){
        return $dbh->{mysql_insertid};
        }
    if ($rv && $sql[0] =~ /^select.*/){
        while ( my $href = $sth->fetchrow_hashref() ) {
            push @rows, $href;
            }
        return \@rows ;
        }
}
```

Table 4-4: common.pl

```
connect logs;

# Top 10 Alerts from alerts table (Field  - alertsign)
select alertsign,count(*) as val from alerts group by alertsign order by val desc
limit 10;

# Top 10 Destination IPs from scans
select dstip,count(*) as val from scans group by dstip order by val desc limit 10;
```

```
# Union of Top 5 from alerts and scans
(select srcip,count(*) as val from alerts group by srcip order by val desc limit 5)
union (select srcip,count(*) as val from scans group by srcip order by val desc
limit 5);

# Create a temporary table using the results of select command
# Temporary tables are dropped when the connection is closed.
create temporary table t1 select srcip,count(*) as val from alerts group by srcip
order by val desc limit 5;

# To compare common srcips between alerts and logs use the next 3-sql
commands
#  Create temporary tables privilege is needed; Temp tables are available only
for that connection.
# Same temp table names can be used from two different connections
1. Create  a temporary table t1 with unique source ips from alerts
create temporary table t1 as select srcip,count(*) as val from alerts group by srcip
order by val desc;
2. 1. Create  a temporary table t2 with unique source ips from scans
create temporary table t2 as select srcip,count(*) as val from alerts group by srcip
order by val desc;
3. Select all the common IPs between the two tables; Careful with large
databases
select t1.srcip from t1,t2 where t1.srcip=t2.srcip;  OR
create table commonsrcip as select t1.srcip from t1,t2 where t1.srcip=t2.srcip;

All the commands can be run in batch mode.
Syntax: # mysql <  'commands.txt'  > 'outputfile.txt'
```

Table 4-5: Some SQL commands used during analysis