



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Network Monitoring and Threat Detection In-Depth (Security 503)"
at <http://www.giac.org/registration/gcia>

Summary

This is the submission for the GCIA version 3.3 practical and consists of three assignments.
The three assignments are listed below:

[Assignment -1 Evolution of IDS](#)

This is a case study of the One Secure Intrusion Prevention Systems and how Network Intrusion Detection Systems are no longer being seen as a passive devices. The report is based on One Secure's IDP version 1.6, before it was acquired NetScreen.

[Introduction](#)

[One Secure Intrusion Detection and Prevention System](#)

[Testing OneSecure IDP](#)

[Tools used for the testing](#)

[Results](#)

[Summary](#)

[References](#)

[Assignment -2 Three network detects](#)

[Detect 1](#) – Fragmentation Scanning, was posted at Incidents.org for public scrutiny

[Detect 2](#) – Shell code, Buffer overflow attack

[Detect 3](#) - Linux Slapper worm

[Assignment -3 Analyze This](#)

This part of the practical examines log files collected from <http://www.incidents.org/logs>.

Here we an attempt is made at describing the network traffic seen as to why it may have happened.

The author tries to make recommendations where possible on how to improve the situation at hand. Hosts the look like they are involved in suspicious activity are also pointed out for further investigation.

[Executive Summary](#)

[Summary of Alerts](#)

[Top 10 Scanners](#)

[Top 10 Destinations](#)

[Top 10 Destination Ports](#)

[Top 10 Sources](#)

[Summary](#)
[Appendix](#)
[References](#)

Assignment-1 [\(To Top of Document\)](#)

Evolution of IDS - One Secure Intrusion Prevention System

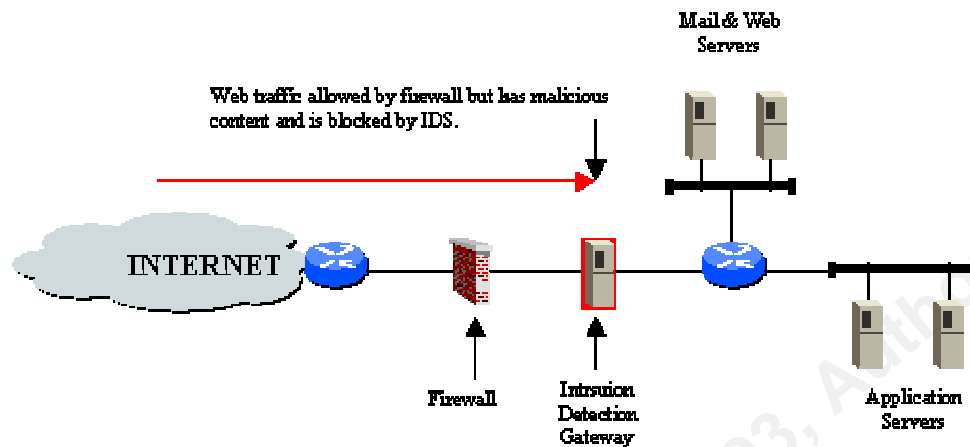
Introduction:

Intrusion Detection Systems (IDS) have been around for a while now, since their conception they have been turned into a valuable commodity in defending information assets of all types of institutions. Traditionally they came in two flavours, Host based Intrusion Detection Systems (HIDS) and Network based Intrusion Detection Systems (NIDS).

HIDS have developed from just monitoring file system changes and system utilization to now being IP stack aware with personal firewalls, monitoring both the network activities and systems activities.

During the last couple years NIDS have become more prominent due to the outbreaks worms (automated exploits that take advantage of network based services) and a rise in the level computer based crime. While Firewalls and VPNs sit on the perimeter of networks restricting most inbound and outbound traffic they still provide small holes that can be penetrated. These holes could be public web servers that sit on the DMZ that could be used as a staging point into the internal network. While this access can be restrictive not many of these devices can actually be content aware thus the idea of NIDS. NIDS provides the capability to monitor the content of the unencrypted traffic that comes into our protected networks. At best NIDS were only passive devices telling the analyst of what he/she just happened, later they were able to have more of an active role. The NIDS system could interface with other network devices like firewalls or routers to either to block the access to the "would be" intruder or to actually attempt to terminate connection by sending a TCP reset to both ends of the connection. Both of these methods these methods have problems associated with Network Latency in that you can't guarantee the intruder will be blocked or the connection will be terminated in time.

People in the intrusion detection arena have recognized these issues and have come with the concept of Intrusion Detection Gateways. Traffic must pass through this device before it enters or leaves the protected network. If the packet is malicious it is dropped or simply not passed to the other interface of the Intrusion Detection Gateway.



Some inroads were made with an open source project called [HogWash](#) which was inline packet scrubber based on the open source NIDS called [Snort](#). Unfortunately most commercial companies and government organizations feel uncomfortable with open source software and feel better with commercial products, however until recently there hasn't been many vendors that produce such devices and if they did they were very expensive (~\$100,000 USD) for networks that approach Gigabit speed making it generally out of reach for most smaller organizations. Recently a new player, called [OneSecure](#) has emerged on the market and is competing with companies such as [TippingPoint technologies](#) and [IntruVert Networks](#). OneSecure developed a product called an Intrusion Prevention System that entered the market for \$16,500 USD⁽¹⁾. This technology now becomes more attainable for the not so large organizations. Recently OneSecure was acquired by [Netscreen](#) so pricing might not remain the same, but for the remainder of this document it will be referred to as OneSecure Intrusion Prevention system.

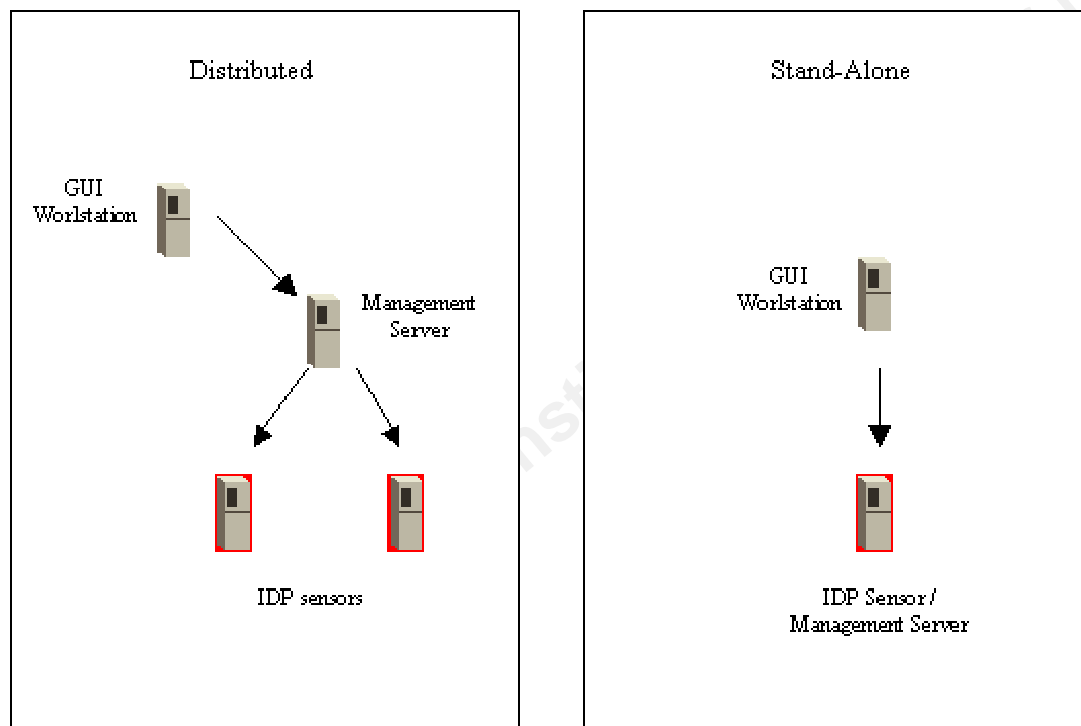
(1) Price taken Network World Fusion <http://www.nwfusion.com/buzz/2002/intruder.html>

This is an Overview of the One Secure Intrusion Prevention System.

One Secure Intrusion Detection and Prevention System [\(To Top of Document\)](#)

Architecture

One Secure Intrusion Prevention System (IDP) can be installed in a Distributed or Stand-Alone configuration that is similar to Checkpoint's Firewall-1. In a distributed model the Management server can be run on a separate platform from the IDP sensor enabling it manage multiple sensors. One Secure recommends that Distributed Model be used in networks that require multiple sensors and bandwidth utilization. The sensors are then configured in a bridging mode so that no changes need to be made to routed networks. There is also one strong advantage of the distributed model and that is when an analyst searching through the logs to get the appropriate view of event can cause high CPU utilization and Disk I/O. This would be detrimental to the performance of the sensor bridging Ethernet packets. The GUI interface is required to run a separate machine in either the Stand-Alone or Distributed configuration.



Sensor

The sensor consists of a Dell Power Edge 1650 and the Intrusion Prevention software that is based on Red Hat 7.2. The sensor can be configured in a high availability or stand-alone mode, below is what options are available with the configuration decision. Although the Bridge configuration is the preferred option, the IDS can be quite flexible on the other ways it can be implemented.

1. High Availability -> Active (inline mode)-> Bridge configuration.
2. Stand Alone -> Passive (using sniffing port or Ethernet Tap)-> Sniffer Configuration
3. Stand Alone -> Active (inline mode) -> Proxy-ARP Configuration
4. Stand Alone -> Active (inline mode) -> Router Configuration
5. Stand Alone -> Active (inline mode) -> Bridge Configuration

An interesting point to note here about their Bridge configuration is that version 1.6 and 2.0 of the sensor can not handle 802.1q trunks in an inline mode. More and more we see the use of VLAN's on switched networks dividing the switch into separate logical network segments while using the same physical device. For redundancy in the switched network we can create trunks, which enable same segments to be seen on multiple switches. Then some one thought of the idea of using one interface of the firewall to terminate separate VLAN's instead of the different interfaces. This also allows you to do a kind of "policy based routing" and also provides more scalable network. Unfortunately if your thinking about putting this One Secure IDP sensor on the trunk in between the firewall interface and the switch,.... it will not work.

Management Server

The management server if it is not running on the sensor can run on a dedicated RedHat 7.2 or Solaris (7 or 8) system. The management server is where the policies are stored and signatures are kept and deployed it also provides a version control system on the policies enabling you to view what has changed and when it was changed, I think this is an important feature in any security device. You can also configure the two alerting mechanisms here, which are email interface and a SNMP interface. There is also a syslog facility that can manage a collection of syslog data. The syslog facility seems like the only way that you can monitor the general health of the sensors and management server for changes in system processes specific for the IDP processes without a GUI being open to the management server all the times.

GUI Interface

The GUI runs on RedHat 7.2 or Win32 platforms. It provides the user interface for the system administration, viewing of the logs, creating reports, system health, signature maintenance, policy creation and policy deployment. The Policy creation is very like the similar to Checkpoint's Policy editor and is very intuitive. Here you can set various parameters for each type of attack like drop connection, close connection, alert on sessions, or capture session data. The payload is reasonably easy to extract from the GUI Interface if this option is set in the policy other IDS. Version 1.6 was OPSEC compliant, which meant you

could interface into the Checkpoint product suites, however this OPSEC compliance has been taken out in version 2.0 due to copyright infringements between Net Screen and Checkpoint.

An advantage of the Policy editor is that it allows to filter in the traffic that you want to see rather than implement all the rules and then write exceptions of what you don't want to see. This is a powerful feature that reduces the signature tuning time.

Diagram of GUI Interface from version 2

© SANS Institute 2003, Author retains full rights

Log Viewer - NetScreen IDP Manager

File Edit View Server Tools Help

Dashboard Log Viewer Security Policies EnterprisePolicy TestPolicy Objects Device Monitor Reports Log Investigator

flag	alarm	time received	attack	source address	src port	destination address	dst port
		7/3/02 6:05:00 PM	HTTP:REGERR:INV-HTTP-VER	64.81.101.100	2056	Corporate Web Server	http
		7/3/02 8:21:00 PM		64.209.168.239	0	Internal Network	0
		7/3/02 9:20:00 PM	MP3:NAPSTER:LOGIN-7777	Internal Workstation	0	100.100.100.100	0
		7/3/02 11:10:00 PM		10.0.109.100	0	10.0.102.101	0
		7/3/02 11:10:00 PM		10.0.109.100	0	10.0.102.101	0
		7/3/02 11:13:00 PM	SCANNER:NMAP:FINGERPRINT	10.0.109.100	ICMP	10.0.102.101	echo-request
		7/3/02 11:16:00 PM	SCANNER:NMAP:XMAS	10.0.109.100	56520	10.0.102.101	http
		7/3/02 11:17:00 PM		10.0.109.100	0	10.0.102.101	0
		7/3/02 11:19:00 PM		10.0.109.100	0	10.0.102.101	0
		7/3/02 11:21:00 PM		10.0.109.100	0	10.0.102.101	0
		7/3/02 11:23:00 PM		10.0.109.100	0	10.0.102.101	0
		7/3/02 11:25:00 PM		10.0.109.100	0	10.0.102.101	0
		7/3/02 11:26:00 PM	SCANNER:NMAP:FINGERPRINT	205.188.160.123	ICMP	Primary Public DNS Server	echo-request
		7/3/02 11:28:00 PM		10.0.109.100	0	10.0.102.101	0
		7/3/02 11:30:00 PM		10.0.109.100	0	10.0.102.101	0
		7/3/02 11:31:00 PM		10.0.109.100	0	10.0.102.101	0
		7/3/02 11:32:00 PM	TCP:AMBIGUOUS:OPT-UNSUP	64.81.101.234	62027	Europe Email Server	http
		7/3/02 11:33:00 PM	SCANNER:NMAP:XMAS	10.0.109.100	62028	10.0.102.101	http
		7/3/02 11:35:00 PM		207.171.187.17	0	10.0.102.101	0
		7/3/02 11:36:00 PM		10.0.109.100	0	10.0.102.101	0
		7/3/02 11:40:00 PM	IP:OPTERR:INVALID-INFRAGMENT	10.0.109.100	0	10.0.102.101	0
		7/3/02 11:41:00 PM		10.0.109.100	0	10.0.102.101	0
		7/3/02 11:42:00 PM	DNS:OVERFLOW:Linux-ADMV2-TCP	64.81.101.100	0	Primary Public DNS Server	0
		7/3/02 11:43:00 PM		10.0.109.100	0	10.0.102.101	0
		7/3/02 11:45:00 PM	SCANNER:ANY:NULL	64.12.149.23	34503	10.0.102.101	http

Summary All Fields

DNS Buffer Overflow x86 Linux (ADMv2) (TCP)

Look for [.*\x89f7 29c7 89f3 89f9 89f2 ac3c fe\x.*] in [stream]

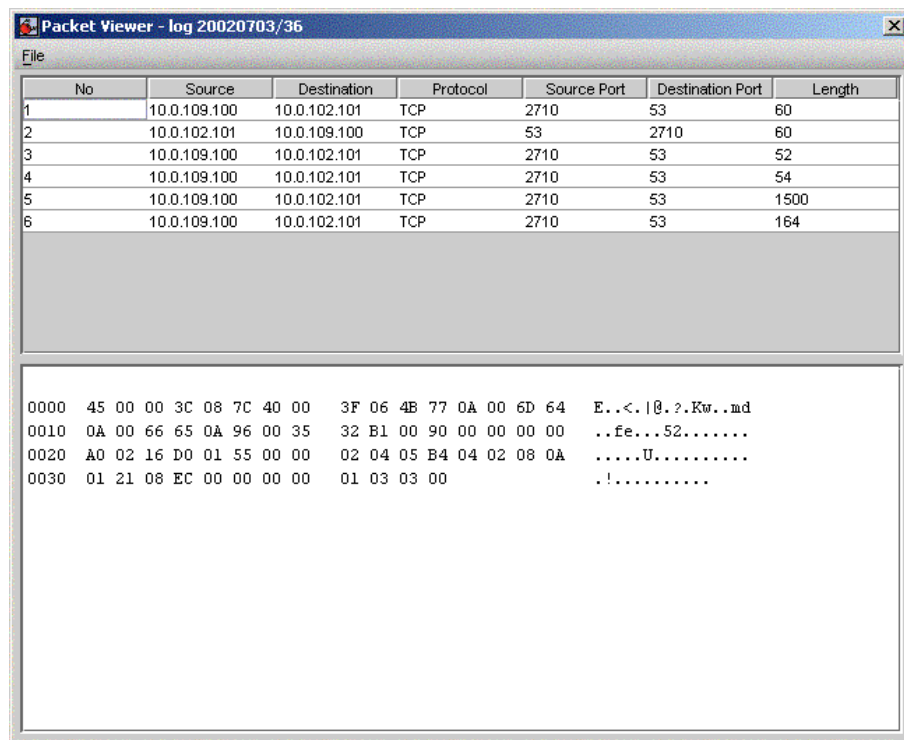
This signature detects attempts to create buffer overflows on an x86 system running Linux. Attackers may send maliciously crafted packets to TCP/53 to overflow the buffer and gain root access.

Packet Data...
Packet Data in External Viewer...
Security Policy...
Attack in Security Policy...
Attack...

Filter
Find
Flag
Show
Hide Log
Unhide Log

Server: demo mode

The Next Diagram is the packet view form the above picture



The image shows a screenshot of a 'Packet Viewer' application window. The title bar reads 'Packet Viewer - log 20020703/36'. Below the title bar is a menu bar with 'File'. The main area is divided into two sections. The top section is a table with 7 columns: 'No', 'Source', 'Destination', 'Protocol', 'Source Port', 'Destination Port', and 'Length'. It contains 6 rows of data. The bottom section is a large text area displaying a hex dump of a packet. The hex dump shows four lines of data, each with a hexadecimal value and its corresponding ASCII representation. The ASCII representation is 'E..<.|@.?.Kw..md'.

No	Source	Destination	Protocol	Source Port	Destination Port	Length
1	10.0.109.100	10.0.102.101	TCP	2710	53	60
2	10.0.102.101	10.0.109.100	TCP	53	2710	60
3	10.0.109.100	10.0.102.101	TCP	2710	53	52
4	10.0.109.100	10.0.102.101	TCP	2710	53	54
5	10.0.109.100	10.0.102.101	TCP	2710	53	1500
6	10.0.109.100	10.0.102.101	TCP	2710	53	164

0000	45 00 00 3C 08 7C 40 00	3F 06 4B 77 0A 00 6D 64	E..<. @.?.Kw..md
0010	0A 00 66 65 0A 96 00 35	32 B1 00 90 00 00 00 00	..fe...52.....
0020	A0 02 16 D0 01 55 00 00	02 04 05 B4 04 02 08 0AU.....
0030	01 21 08 EC 00 00 00 00	01 03 03 00	..!.....

The next view is of the policy manager

Security Policies - NetScreen IDP Manager							
File Edit View Policy Server Tools Help							
Traffic Anomalies SYN-Protector Network Honeypot Main Backdoor Detection Sensor Settings							
No.	Match			Look For		Action	
	Source IP	Destination IP	Terminal	Attacks		Notification	Insta
1.	Security Auditing Group	any	<input checked="" type="checkbox"/>	— none	none	— none	any-sensor
2.	any	Corporate DMZ Network	<input checked="" type="checkbox"/>	Critical: DNS Attacks Critical: HTTP Attacks Critical: SMTP Attacks more...	drop connection	logging alarm run script more...	Corporate D
3.	Internal Network	Internal MP3 Server Group	<input type="checkbox"/>	Info: MP3 and File Sharing Attacks and...	none	logging session	Europe IDP S Internal IDP :
4.	any	any	<input type="checkbox"/>	Severity 2 (High)	drop connection	logging alarm	any-sensor
5.	any	Primary Public DNS Server Secondary Public DNS Server	<input checked="" type="checkbox"/>	High: DNS Attacks Medium: DNS Attacks	close server	logging log packets(5/10)	Corporate D Europe DMZ
6.	any	Corporate Email Server Europe Email Server	<input type="checkbox"/>	High: SMTP Attacks Medium: SMTP Attacks	close server	logging alarm session	Corporate D Europe DMZ
(disabled)	any	Corporate Email Server Europe Email Server	<input checked="" type="checkbox"/>	SMTP-Context:Confidential	none	logging	Corporate D Europe DMZ
8.	any	WEB Server Group	<input checked="" type="checkbox"/>	High: HTTP Attacks Low: HTTP Attacks Medium: HTTP Attacks	close client & server	logging session	Corporate D
Policy: EnterprisePolicy							

Testing OneSecure IDP [\(To Top of Document\)](#)

Before implementing any new technology is always wise to see how the technology operates in a test environment. This is so that you can verify that the product works as documented and can be implemented in the way you think it whilst not producing any unexpected behavior.

Test Setup Prelude

Our Primary interest in this device was to use it in a bridging mode so that we could insert the device with little or no effort put into changing the existing network infrastructure. The main desire to use this device came from a need to complement our stateful firewall architecture and give it the application awareness that it lacks whilst maintaining the flexibility that this type of firewall provides. Another the reason to use this device in a bridging mode is that it is the only mode where high availability can be achieved and that there are no single points of failure.

We used the Open Source Security Testing Methodology Manual as a guide on how to test this device. This is available from <http://www.ideahamster.org>. It also important to take into account the results produce by readily available tools, as one person's interpretation may be different from another. Please refer to Appendix A "Notes about the tools used".

Test Equipment

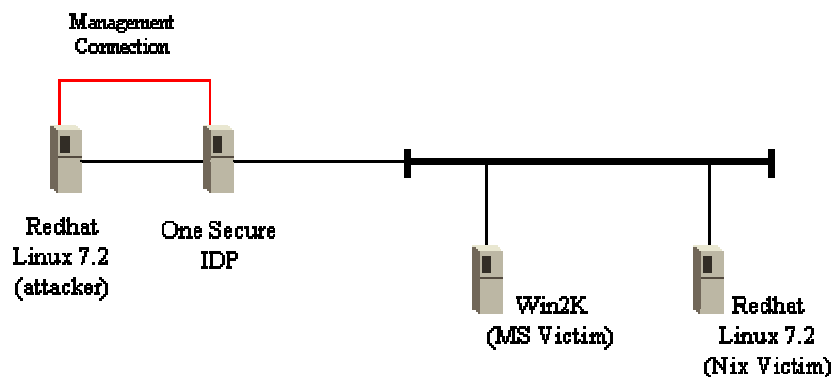
As it was I only has limited I only limited test equipment available in which to do some testing to see if worked they way we believed it did.

- 100 Mb Switch
- 2 x Crossover Cable
- 3 x Straight through Cables
- HP E60 running RedHat 7.2 used as an attacking host and IDP management station
- HP E60 running RedHat 7.2 with every service running for a Victim.
- HP E60 running Win2K Server with all service running.
- OneSecure IDP system
- Distribution media for RedHat 7.2
- Distribution Media for Windows 2000 Server

Test Network

Time on all systems where synchronized with NTP so that common frame of reference could be used when interpreting the results produced. The Source timeserver that was used was the attacking host.

Test Network



Attacking Host

The Attacking host also runs the GUI interface to the IDP system, was setup using two crossover cables. One Cable to connect to the management interface of the IDP system and the other was to launch the attacks from. RedHat was installed with all the libraries and compilers so that third party tools could be compiled and run. The following is a list of the tools installed.

- Sneez, Snot and Stick used for generating traffic for the effect of false positives.
- Whisker, Nmap and Nessus used for system scanning, vulnerability assessment but also to test the IDS using evasion techniques
- Fragroute, Nemesis and Tcpreplay to generate fragmented traffic also for the purpose of IDS evasion

Nix Victim

This was an unpatched version of RedHat 7.2 with every possible service running. So that a base line could be created and results could be obtained as to what was allowed before the IDP's countermeasures were enabled.

MS Victim

This was an unpatched version of Win2K with every possible service running. So that a base line could be created and results could be obtained as to what was allowed before the IDP's countermeasures were enabled.

One Secure IDP

The OneSecure IDP System was configured as a bridge and inserted between the Switch and the Attacking host. While the management interface was connected to a separate interface of the Attacking host. The IDP system also comes with TCP dump so see what the traffic looks like as it traverses the device and also compare it with the traffic captured on the attacking host and the Victim machines. The signature set that was loaded was the default signature set from the vendor of some 1300 signatures.

Tools used for the testing [\(To Top of Document\)](#)

Sneeze, **Snot** and **Stick** are tools designed for testing IDS robustness these have a common principle in that they use snort attack signatures to build packets and fire them down them across the network. When using these tools it is important to keep in mind the only thing that these really testing are signature recognition and the alerting and logging capability of an IDS and it's underlying components. IDS's that are configured/designed to be stateful may generate no events of the "attacks" sent across on the wire.

If the testing of signature recognition, alerting in logging of the IDS is the objective of test it maybe required that the "stateful" features be disabled or another means to test this functionality derived.

Below tcpdump was used to capture the output from some of these tools to demonstrate how using stateful IDS's that the results may appear to be negative.

Sneeze Capture:

```
tcpdump -nlr sneeze.dmp -vv port 17830.
```

```
18:13:44.371088 127.0.0.1.17830 > 192.168.10.3.http: . [tcp sum ok] 0:21(21) ack 0 win 65535 (DF) [tos 0x10] (ttl 64, id 0, len 61)
```

Stick Capture:

```
tcpdump -nlr stick.dmp -vv 'tcp and port 80 and port 9329'
```

```
18:34:37.372757 192.168.10.3.9329 > 192.168.10.3.http: P [bad tcp cksum 5eae!] 561915240:561915380(140) ack 3232238083 win 38726 (ttl 252, id 30243, len 180)
```

Snot Capture:

```
tcpdump -nlr snot.dmp -vv 'tcp and port 80 and port 18328'
```

```
20:01:28.763744 192.168.10.8.18328 > 192.168.10.3.http: P [tcp sum ok] 1184600576:1184600601(25) ack 4244380285 win 49693 (ttl 76, id 28821, len 65)
```

Here I have used a "bpf" filter with Tcpdump to demonstrate the missing Syn and Syn/Ack from a normal TCP connection. If the IDS were stateful than it should not have any connection information about these packets and should ignore them.

Whisker, **Nmap** and **Nessus** are scanning tools used to look at hosts on a network and check them for vulnerabilities. Malicious hackers to do security audits of your sites for you with these commonly used tools.

Nmap can be used in a stealthy mode to identify operating systems and available services that may be available on certain hosts. The idea to use such a tool is to see the resolution if the IDS and to check if the IDS is able to detect long slow port scans.

Whisker is used for scanning of web servers to gain information about what type of web server is running and any vulnerable scripts that maybe running on a particular web server. It tries to hide its activities by using different types of URL encoding in order to slip by an IDS that cannot normalize the web traffic.

The use of this tool in an IDS test is to check whether the IDS can normalize web traffic and its signature base for different web servers.

Nessus is more of a modular tool that can incorporate other open source tools such as Nmap in which to conduct an audit of the state of security of your site.

Essentially it can fingerprint the host check what services are running and if those services are vulnerable to certain attacks.

All these scanners have characteristic fingerprints and features to mask their activity, some IDS's can identify them, some can't, at the very least the IDS should be able to pick up any sort of scanning activity as this is usually the precursor for an actual attack.

Fragroute, Nemesis and TcpReplay are packet-crafting tools or modify traffic streams in order to avoid detection.

Fragroute initially written by Dug Song and it's theory is based upon the paper written by Secure Networks "[Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection](#)" paper of January 1998. Once fragroute is setup the normal attack passes through it onto the target host this traffic is then fragmented in various ways. In this situation the IDS is then tested to see if it can reassemble fragmented traffic that may come out of order or be overlapping.

Nemesis is a packet crafting command line tool which makes it ideal to for scripting attacks, "Nemesis attacks directed through fragroute could be a most powerful combination for the system auditor to find security problems that could then be reported to the vendor(s), etc." - Curt Wilson in Global Incident Analysis Center Detects Report (SANS Institute - Nov 2000).

TcpReplay enables you to replay previously recorded traffic from Tcpdump or Snoop across the network. This is good when you look at the inherent problems that are associated with tools like Sneeze, Snot and stick when testing stateful IDS's being able to send prerecorded traffic through tools like fragroute would give the IDS a real workout.

Libnet and Net-RawIP provides a common high-level API for packet creation and packet injection across most current operating systems. This is prerequisite for some of above-mentioned tools

Results [\(To Top of Document\)](#)

Type of IDS

One Secure Intrusion Detection and Prevention System (IDP) is an Intrusion Detection System that has the capability of being used in an inline mode, almost like an application firewall.

IDS performance under heavy load.

Not tested packet generation, as tools to produce over a 100MB were not available to us although the manufacturer claimed that it could do Gigabit speed.

Type of packets dropped or not scanned by the IDS
Encrypted Traffic

Type of protocols dropped or not scanned by the IDS

IDS only scans for or is stateful for TCP, UDP and ICMP traffic. It can also detect an IP Protocol scan.

Note of reaction time and type of the IDS

Due to the nature of this IDS being inline it does not suffer from the problems associated with normal IDS and behaves similar to an application firewall, If an attack occurs it is dropped instantly rather than trying to send a TCP reset to the end points of the connection, or experiencing network latency while reconfiguring firewall rules

Note of IDS sensitivity

Sensitivity is a configurable parameter, default out of the box the it detects most fast scans not spread out longer than a 4 events over a 120 second period, this is also a tunable parameter. Some IDS evasion techniques were successful when used against this machine like URL encoding or session splicing. However the amount of false positives generated when using tools like snort and stick are low due to the stateful signature engine. Packets built by these tools are put together from a [Snort](#) Signature and sent to the target host with out actually completing a 3-way handshake for TCP. False positives are easier to create with UDP traffic due to UDP being connectionless. The Signature set that comes with the default install is also not very large, this could also attribute to the low number of events being generated.

Rule map of IDS

Rules from this IDS cannot be exported but are ASCII text files located in a directory. The layout of these files are similar to the inspect script and Objects.C from [Checkpoint's](#) of Firewall-1.

The Policy is GUI is also almost the same as Checkpoint Firewall-1 but has the added feature of version control so that you can see previous versions and what was changed

IDS false positives

False positives are events that are non malicious but were detected by the IDS as malicious traffic.

The list of false positives for TCP traffic was low but this could also attributed to the tools being used. Snort Stick and Sneeze build the "Bad Packet" from a Snort Signature file and sends the packet across the wire. What these tools fail to do is build a TCP connection with a three-way handshake.

Out of these tools Snort-092a was used, as it seemed to build packets with out any checksum errors. The purpose of a checksum is to detect if any modifications that may have occurred in the protocol header and/or data in transit, If there has been, the packet will then be discarded by the receiving host. I would think that any IDS system worth its metal would also discard this type of traffic hence the desire to use a tool that does not produce these errors.

Snort-092a used the same signature set from stick called vision.txt which contains 1072 signatures of these 145 are UDP related. The UDP related attacks were taken out of the vision.txt file so that UDP and TCP/IP could be tested separately against the OneSecure box and also to see how it would handle the different protocols

The OneSecure IPD system produce no false positives for TCP when snort was used to generate the hostile traffic, only producing port scan related events. The port scan events can probably be attributed to the high volume of traffic coming out of the attacking host directed against the Linux victim. OneSecure has a state table that it uses in stateful analysis of attacks. Without a proper handshake taking place it ignored these crafted packets. The same was observed for UDP traffic only producing an alert for a "DeepThroat, Server status Client Request", It also handle UDP in a stateful manner creating a virtual entry for UDP traffic similar to firewall-1.

False Negatives

False negatives are events the IDS should have picked up but did not.

The context of this results needs to be taken into account when reviewing the IDS. For example the tool being used, Whisker in this case, could have formed a perfectly valid request to the web server that is not considered to be an actual attack but a legitimate way of requesting the data.

Whisker1.4 was used with the standard scan database to probe the Web servers with 10 IDS evasion Techniques that it has and the results recorded. Below is what evasion techniques used by whisker the OneSecure box did not produce and event for:

- URL encoding
- Null Method
- Tab Separation
- Case sensitivity
- Windows delimiter
- Session splicing

IDS missed alarms

A missed alarm is when something happens to the sensor that may affect its normal operating mode, e.g. a process that has stalled. When testing the device we had the manager and IDS on the same system. The manager itself monitored all the processes running on the IDP Platform and created an alert when one processes failed on the GUI console.

It did not appear that you could that you could monitor the IDS sensor itself with a third party product like HP Open View, as it was not documented nor did we have a third party platform like HP Openview to verify this

Ability to handle fragmented traffic

Good although the default settings were changed to reassemble fragmented traffic. With this set the OneSecure device either reassembled the fragmented traffic if it could while and then proceeded to check the contents, or dropped the traffic if the fragmentation made no sense while logging that it had done so.

Limitations

Due to the lack of equipment where unable to verify that the IDP system could operate at Gigabit speeds.

We were also unable to verify its integration with other vendors such as Checkpoint or test its protocol anomaly detection capabilities.

Summary [\(To Top of Document\)](#)

The OneSecure Intrusion Detection and Prevention System offer's advances in over current accepted IDS systems in the way it inspects network traffic to reduce false positives and produce more accurate detects. It also having an inline offers significant improvement from traditional passive systems in that you can now see what it just stopped with a higher guarantee for success. This buys time for incident response to make sure the targeted systems have been patched rather than having to do forensics work and damage assessment.

It appears that once there were application firewalls where they had specific proxies for limited protocols. Then came the stateful firewalls offering more flexibility with newer Internet protocols but also producing weakness in the network perimeter. Now with inline IDS's like OneSecure these weaknesses in the network perimeter are now getting less and less. The next 12 months will prove to be very interesting as these I think the technologies of the stateful firewalls and inline IDS will merge into one device.

References: [\(To Top of Document\)](#)

- Herzog, Pete. Open Source Security Testing Methodology Manual v2.0 2002. October 18 2002
<http://www.ideahamster.org/download.htm>
- W. Richard Stevens, TCP/IP Illustrated Volume 1 December 8, 2002
- Secure Networks, Inc. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection January 1998. October 20 2002
<http://secinf.net/info/ids/idspaper/idspaper.html>
- HogWash, an inline packet scrubber based on Snort. December 2, 2002
<http://hogwash.sourceforge.net>
- Marty Roesch, Snort an Open Source Network Intrusion Detection System. December 2, 2002
<http://www.snort.org>
- NetScreen, Intrusion Detection and Prevention, formally OneSecure December 2, 2002
<http://www.netscreen.com/products/idp.html>
- Phoneboy –inspect script
<http://www.phoneboy.com>
- **Libnet** - provides a common high-level API for packet creation and packet injection across most current operating systems.
OS: Unix
Homepage:
Source Download:
<http://www.packetfactory.net/libnet/>
- **Net-RawIP-0.09d** – a perl module for creating raw IP packets
OS: Unix

Homepage: <http://www.cpan.org>

Source Download:

<http://www.cpan.org/authors/id/S/SK/SKOLYCHEV/Net-RawIP-0.09d.tar.gz>

- **fragrouter** - Fragmenting packets to evade IDS.

OS: Unix

Homepage: <http://www.anzen.com/research/nidsbench/>

Source Download:

<http://the.wiretapped.net/security/packet-construction/fragrouter-1.6.tar.gz>

- **nemesis** - Generating / spoofing various packets.

OS: Unix

Homepage: N/A

Source Download: <http://the.wiretapped.net/security/packet-construction/nemesis/nemesis-1.32.tar.gz>

- **nessus** - Triggering scanning alarms.

OS: Unix

Homepage: <http://www.nessus.org/>

Source Download: <http://www.nessus.org/download.html>

- **nmap** - Slow scanning attempting to "fly under the radar".

OS: Unix

Homepage: <http://www.insecure.org/nmap/index.html>

Source Download: <http://download.insecure.org/nmap/dist/nmap-2.54BETA30.tgz>

- **sneeze** - Testing Snort alarm and logging capability.

OS: Unix

Homepage: N/A

Source Download: <http://snort.sourceforge.net/sneeze-1.0.tar>

- **snot** - Testing IDS robustness, as well as alarm and logging capability.

OS: Unix

Homepage: <http://www.sec33.com/sniph/>

Source Download: <http://www.sec33.com/sniph/snot-0.92a.tar.gz>

- **stick** - Testing IDS robustness, as well as alarm and logging capability.

OS: Unix

Homepage: <http://www.eurocompton.net/stick/>

Source Download: <http://packetstormsecurity.org/distributed/stick.tgz>

- **tcpreplay** - Replaying real traffic in which to hide attacks.

OS: Unix

Homepage: <http://www.anzen.com/research/nidsbench/>
Source Download: <http://www.anzen.com/research/nidsbench/tcpreplay-1.0.1.tar.gz>
- **whisker** - Triggering URL alarms or attempting to slip obfuscated URLs past IDS.
OS: Unix
Homepage: <http://www.wiretrip.net/rfp/>
Download: <http://www.wiretrip.net/rfp/bins/whisker/whisker.tar.gz>

Assignment-2 [\(To Top of Document\)](#)

Three Network detects

Detect 1- Fragmentation Scanning (Posted at incidents.org) [\(To Top of Document\)](#)

Log file 2002.7.24
[**] [1:1322:4] BAD TRAFFIC bad frag bits [**]
[Classification: Misc activity] [Priority: 3]
08/24-11:52:42.184488 192.9.100.88 -> 138.97.10.219
TCP TTL:232 TOS:0x0 ID:0 IpLen:20 DgmLen:40 DF MF
Frag Offset: 0x0458 Frag Size: 0xFFFFFBBC

Log file 2002.7.24
11:52:42.184488 192.9.100.88 > 138.97.10.219: (frag 0:20@8896+)

Log file 2002.7.25
04:45:09.524488 192.9.100.88 > 138.97.222.130: (frag 0:20@9000+)

Log file 2002.7.26
12:12:45.864488 192.9.100.88 > 138.97.77.28: (frag 0:20@8896+)
18:40:27.064488 192.9.100.88 > 138.97.144.41: (frag 0:20@34232+)

Log file 2002.7.30
03:51:08.954488 192.9.100.88 > 138.97.129.193: (frag 0:20@10104+)
04:04:17.864488 192.9.100.88 > 138.97.126.77: (frag 0:20@8360+)
10:06:18.274488 192.9.100.88 > 138.97.196.88: (frag 0:20@32952+)
18:40:39.074488 192.9.100.88 > 138.97.44.183: (frag 0:20@8896+)

Source of Trace:

The source of this event came from the logs for GIAC certification located here:

<http://www.incidents.org/logs/Raw/2002.7.24>

<http://www.incidents.org/logs/Raw/2002.7.25>

<http://www.incidents.org/logs/Raw/2002.7.26>

<http://www.incidents.org/logs/Raw/2002.7.30>

Detect Generated by:

The Detect was generated by Snort Version 1.9.0 (Build 209) in read back mode of a tcpdump binary log by the following signature:

alert ip \$EXTERNAL_NET any -> \$HOME_NET any (msg:"BAD TRAFFIC bad frag bits"; fragbits:MD; sid:1322; classtype:misc-activity; rev:4;)

The signature looks at traffic flowing from an outside network into an internal network with both the (M) More fragments and (D) Don't fragment flag set in an IP packet. This is not necessarily illegal traffic but is more to highlight the fact the fragmentation is taking place. It is up to the analyst to decide if it is malicious or not.

Probability the Source Address was Spoofed:

Low to medium, this is a stimulus, and the Attacker would be looking for a response to this packet in the form of an ICMP "time to live exceeded: fragmentation timeout" message. For the connection to be successfully spoofed the attacker would have to be able to intercept the return traffic that may be generated from this activity.

Attack Description:

Fragmentation scanning, the attacker is sending a crafted packet that looks like fragmentation to illicit a response from a possible host is up or not.

Attack Mechanism:

Note the following:-

- The Ethernet MTU size is 1500 Bytes
- The Point-to-Point protocol has an MTU size of 296 bytes. This is the smallest "normal MTU"

This type of fragmentation is illegal as a nonterminal fragment sizes should be in multiples of 8. That means all packets in a fragment chain should be divisible by 8, except for the last one. The last packet will not have the MF flag set. The MF flag here is indicated by the "+" sign in the TCPDump.

Every packet that leaves a host has an IP identification number. This number incremented by one for each consecutive packet leaving the host. In TCPDump tool, the fragment id seen in the output "(frag 0:20@8896+)" is the IP id. The packet is 40 bytes long, the IP header is 20 bytes, the 4th and 5th byte being 0000, which equates with the fragment id of 0.

```

11:52:42.184488 192.9.100.88 > 138.97.10.219: (frag 0:20@8896+) (ttl 232, len 40, bad cksum ff99!)
0x0000  4500 0028 0000 6458 e806 ff99 c009 6458      E..(..dX.....dX
0x0010  8a61 0adb 0ef4 0050 0267 1928 0267 1928      .a.....P.g.(.g.(
0x0020  3e04 0000 81a0 0000 0000 0000 0000      >.....

```

The fragment ID of zero is not common, but to have it from multiple times from the same source would be even more rare. Although, Linux does have an IP id of zero more frequently than most operating systems.

Before Assuming any thing lets look at what we do and don't know: We see one alert from this host with only the binary data for this event in the first log and no other log entries. We don't have a complete traffic log, so we can't see all the traffic from this host. We don't know location of this snort sensor, so we don't know if it was behind a firewall or not. We do know that it was generated by Snort, but do not know what rules are loaded so we can't tell if snort will saw the response or not.

Given that this packet has a high probability of being crafted, this means every field in the packet could be crafted including the offset and the protocol field. If a receiving host was to receive a packet like this, with out the first packet in the fragment chain, the receiving host would send an ICMP "time to live exceeded:fragmentation timeout" message if it is up or no response if is down.

Correlations:

<http://www.nwfusion.com/newsletters/sec/0906sec2.html>
http://www.insecure.org/nmap/nmap_doc.html

Sendip, hping2 and Nemesis are more likely a tools which would allow you to craft a packet of this nature.

<http://freshmeat.net/projects/sendip/>

Evidence of Active Targeting:

There is no other type of activity from 192.9.100.88 over the following days for this occasional fragmented traffic directed to this host and others. This could be part of larger network mapping effort, but there is no there evidence seen by the public from this host, bearing in mind that these IP addresses have been munged. So I would say this maybe an attacker reconnaissance phase actively looking for targets on this network.

Severity:

The formula used to rank the severity of the incident is as follows:

$$(\text{Target Criticality} + \text{Attack Lethality}) - (\text{System Countermeasures} + \text{Network Countermeasures}) = \text{Attack Severity}$$

Each element is ranked 1 to 5, 1 being low, 5 being high. The maximum score (which is also the worst-case scenario) is 8. The minimum score (which is the best-case scenario) is -8.

Target Criticality: 5

Nothing is known about the target system and nothing is known about the service being provided by the target system.

Attack Lethality: 2

This is an information probe.

System Countermeasures: 1

If this traffic made it to the destination, you could not do much than wait for the other packets to arrive, you may be able to set what icmp error messages are returned.

Network Countermeasures: 1

Most modern firewalls would reassemble the traffic before letting it pass to the internal network.

Attack Severity: $(5+2)-(1+1)=5$

Defensive Recommendations:

Implement perimeter defense that does not actually allow abnormally fragmented traffic to pass, like a modern firewall or an inline IDS system.

Multiple Choice Question:

What BPF filter would work to see only the fragmented traffic from a TCPDump

log:

- a) $ip[6] \& 32 \neq 0$
- b) $tcp[13] \& 0x13 \neq 0$
- c) $ip[10] = 0x17$
- d) $ip[9] = 0x17$

Answer (a) do logical AND on the 6th byte from the IP header with a decimal of 32 would mask out the (M)ore fragments flag indication fragmentation is taking place.

Questions from Incidents.org

Question: Andrew Rucker Jones (The comments he posted to me caused me to redefine my initial analysis from Insertion Attack to Fragmentation scanning).

I'm not an expert, but can i offer a different opinion? Your tcpdump filter catches ALL traffic from 192.9.100.88, so unless You have not included all of the packets displayed by tcpdump (which i believe You would need to do to make Your argument convincing), this is the only one. That means, it would not overwrite something in a buffer, because there is no buffer for this connection. The fragment ID of 0 also looks very crafted, and not part of an evasion attempt on an existing TCP connection. The reason i say this is that a real TCP connection into which we wished to insert something would in all ikelyhood not have an IP ID of 0. Might i suggest that this could be a mapping attempt? After a certain amount of time, the receiving host will be required to send back an ICMP "time to live exceeded: fragmentation timeout" message, and that would let the attacking host know that the target host exists.

Answer:

Thanks for your comments.

One thing I should have learned from my Mathematics lecturer at University, never ASSUME anything, because you make an ASS out of U and ME.

I should have noticed the packet the ID and the penny should have dropped that it as zero indicating a crafted packet, and of course in a crafted packet you can craft any thing including the protocol field.

I assumed that their must have been 8896 bytes beforehand so at least 6 packets if it was using packet lengths or around 1500 bytes. I didn't take into account that if it was crafted they whole situation could be crafted including the fragment offset all I have is just one packet. And I also thought that if this was a scanning technique why go to so much effort in building a TCP connection to scan a host.

The answer to that is to make the inexperienced like myself to think that it is something else.

Here is the info frome looking at logs after that date:

```
bash-2.05b$ tcpdump -nlr 2002.7.24 'host 192.9.100.88'
11:52:42.184488 192.9.100.88 > 138.97.10.219: (frag 0:20@8896+)
bash-2.05b$ tcpdump -nlr 2002.7.25 'host 192.9.100.88'
04:45:09.524488 192.9.100.88 > 138.97.222.130: (frag 0:20@9000+)
bash-2.05b$ tcpdump -nlr 2002.7.26 'host 192.9.100.88'
12:12:45.864488 192.9.100.88 > 138.97.77.28: (frag 0:20@8896+)
18:40:27.064488 192.9.100.88 > 138.97.144.41: (frag 0:20@34232+)
```

Question. André Cormier

I've never thought of this before, but now that i'm reading this paragraph it jumped right in my face. There is no such thing as a fragment id...Granted that this field is used for reassembly and that the tcpdump MAN page says that it is the fragment id :

tcpdump MAN page :

(frag id:size@offset+)

(frag id:size@offset)

(The first form indicates there are more fragments. The second indicates this is the last fragment.)

Id is the fragment id. Size is the fragment size (in bytes) excluding the IP header. Offset is this fragment's offset (in bytes) in the original datagram.

But i'm not sure it is right to call this field the fragment id (No offense intended to the MAN page authors). It's the IP identification number. It identifies the IP packet not the fragment. It's the offset that identifies the fragment.

RFC 791 says :

Identification: 16 bits

An identifying value assigned by the sender to aid in assembling the fragments of a datagram.

Every packets that leave a hosts has an identification number even if the packet is not fragmented. That also explains why an identification number of 0 is rare since for every IP packet that leaves the host the identification number is incremented. (I suggest you add that as a justification for your statement.

tcpdump shows this field whith the frag information because it's relevant to fragments. You can always show Ip identification numbers by supplying the -v flag.

Any one agrees with me ? ;-)

Answer.

I agree, I checked it out this morning, the packet is 40 bytes long, the IP header is 20 bytes, there is no fragmentation id field in the IP header, only an IP id field. This is the 4th and 5th byte being 0000, which equates with the fragment id of 0

11:52:42.184488 192.9.100.88 > 138.97.10.219: (frag 0:20@8896+) (ttl 232, len 40, bad cksum ff99!)

0x0000 4500 0028 0000 6458 e806 ff99 c009 6458 E..(..dX.....dX

0x0010 8a61 0adb 0ef4 0050 0267 1928 0267 1928 .a.....P.g.(.g.(

0x0020 3e04 0000 81a0 0000 0000 0000 0000 >.....

Question: Ashley Thomas

>> for every IP packet that leaves the host the identification number is incremented.

This is not true always. Take the case of linux O.S. It keeps ID as '0' when the datagram is not a fragment and the DF bit is set. ID value = 0 is rare for a fragmented datagram but not otherwise.

>> It's the offset that identifies the fragment.

IMHO, both ID and offset needs to be considered while identifying/reassembling together with source, destination addresses and protocol.

Answer:

I did not know that so I tried it for myself. The Linux machine is w.x.y.z.

10:30:54.350562 w.x.y.z.34860 > a.b.c.d.22: . [tcp sum ok] ack 2426 win 9280 <nop,nop,timestamp 566769595 285167845> (DF) (ttl 64, id 0, len 52)

But I do not agree entirely with the offset, the offset is used to specify the location of the fragment in the whole chain, I would say it is combination of the frag id and the offset that identifies the fragment.

Detect 2 – Shell code, Buffer overflow attack [\(To Top of Document\)](#)

Snort log:

Nov 27 22:38:50 white-widow snort: [1:1390:3] SHELLCODE x86 inc ebx NOOP

[Classification: Executable code was detected] [Priority: 1]: {TCP}

194.109.137.218:80 -> my.host.com:1026

TCP dump log:

22:38:50.386774 194.109.137.218.http > my.host.com.1026: . 1340632441:1340633889(1448) ack 1337623163 win 7722 <nop,nop,timestamp 606825015 5895106> (DF)

```
0x0000 4500 05dc 26c2 4000 3406 f966 c26d 89da    E...&.@.4..f.m..
.....<snip>.....
0x0130 f380 873c e231 4f78 ca33 feff 2cd6 311b    ...<.10x.3...,1.
0x0140 0200 00a0 e3c6 8686 86b8 8e10 3a17 e272    .....r
0x0150 3a97 bbd0 d111 e242 eee2 bad3 9db8 ee84    :.....B.....
0x0160 dc85 d011 4288 cb5d 4343 4343 4343 4343    ....B..]CCCCCCCC
0x0170 4343 4343 4343 4343 4343 4343 4343 4343    CCCCCCCCCCCCCCCC
0x0180 4343 03cf e39d de1f f02d df9d 0f3e f9e2    CC.....-...>..
0x0190 9b1f 7ef9 e3ff 3776 ff7f 7f06 1962 9811    ..~...7v.....b..
.....<snip>.....
0590  ecb2 c73e 071c 72c4 3127 9c72 c639 175c    ...>..r.1'.r.9.\
0x05a0 72c5 3537 dc72 c73d 0f3c f2c4 332f bcf2    r.57.r.=.<..3/..
0x05b0 c63b 1f7c f2c5 373f fcf2 c7c0 3fa8 3f83    .;|.7?....??.
0x05c0 0c31 cc08 a38c 31ce 0493 4c31 cd0c b3cc    .1....1...L1....
0x05d0 31cf 028b 2cb1 cc0a abac b1ce    1...,.....
```

Source of Trace:

The detect came from my home network which has a dedicated screened subnet

for the purposes of this project. The subnet is protected by stateful firewall based on IPtables and Linux. Traffic enters this network via Masqueraded address.

Detect Generated by:

The Detect was generated by Snort Version 1.9.0 (Build 209)

With the following signature:-

```
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86 inc ebx NOOP"; content:"|43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43|"; classtype:shellcode-detect; sid:1390; rev:3;)
```

The rule is looking for traffic flowing from external network to the internal network that may have machine code for NOOP (No Operations) contained within it. The No Operation instruction set is characteristic of a buffer overflow occurring.

Probability the Source Address was Spoofed:

Low. This is a single frame from a TCP session where the TCP handshake had to have taken place.

Attack Description:

The frame shows a number of hex 43s followed by some ASCII strings.

Attack Mechanism:

NOOP's are usually used to pre-pad code in a buffer overflow attack. A buffer overflow attack takes advantage of a lack of boundary checking in programming

variables. If a variable is only designed to have 64 bytes of data and more than 64 bytes of data is given to the variable, it causes the buffer to overflow.

This overflow data is usually machine code for NOOPs. A NOOP code on a x86 Platform is hex 90s, but can be any machine code that does not perform any

useful operation, like 'mov ax,ax'. In this case we see hex '43 43..' which is the x86 opcode for inc ebx in the dump.

An important thing to note here is that the traffic flow, it is not in the direction of my web server rather from my machine to an external web server.

<http://194.109.137.218> resolves to a debian linux website, more than likely the user was downloading some binary files from here and the '43 43' pattern was seen in this binary transfer.

Also for a buffer overflow to take place you would need a lot more of the '43 43..' in the pattern to overflow the machine's memory.

Correlations:

<http://www.whitehats.com/IDS/181> describes this type of attack.

<http://www.geocrawler.com/archives/3/6752/2002/8/50/9273052/> is a trace captured by someone else.

Evidence of Active Targeting:

None flow of the traffic is flowing away from the web server on the DMZ, this is a false positive

Severity:

The formula used to rank the severity of the incident is as follows:

$$(\text{Target Criticality} + \text{Attack Lethality}) - (\text{System Countermeasures} + \text{Network Countermeasures}) = \text{Attack Severity}$$

Each element is ranked 1 to 5, 1 being low, 5 being high. The maximum score (which is also the worst-case scenario) is 8. The minimum score (which is the best-case scenario) is -8.

Target Criticality: 1

Target was a test web server used for research

Attack Lethality: 1

This was not an attack, it was a false positive.

System Countermeasures:3

It was a current operating system with all the patches applied

Network Countermeasures:3

The if the attack was in the direction of the web server it still would have made it to the server as port 80 is open..

Attack Severity: $(1+1)-(3+3) = -4$

Given the payload it does not appear that any shell commands were executed.

Defensive Recommendations:

The defenses in this case appear to be fine and this is a false positive. The rule could definition for \$SHELLCODE_PORTS could be better so that the it only contains services provided by my network and not any. This hex code can appear in binary data.

Multiple Choice Question:

Is it important that a packet capture of critical events is made,

- A. so it can be determined if the attack is false positive.
- B. so as to test the logging mechanisms.
- C. so that you can perform protocol analysis.
- D. so that you can test disk I/O.

Answer A. It is important to capture the packets so that you definitely know what actually caused the event to trigger. This helps you tune out the false positives as you have a better insight as to what caused the event.

Detect 3 - Linux Slapper worm [\(To Top of Document\)](#)

Jan 18 09:51:52 white-widow snort: [1:1881:2] EXPERIMENTAL WEB-MISC bad HTTP/1.1 request, potential worm attack [Classification: access to a potentially vulnerable web application] [Priority: 2]: {TCP} 62.165.191.132:43132 -> my.host.com:80

TCP Dump from this event:

First part of the attack checking the Apache version which is displayed in the server error.

```
09:51:51.289109 62.165.191.132.43132 > my.host.com.http: S 2429552203:2429552203(0) win 5840 <mss 1460,sackOK,timestamp 673244545
0,nop,wscale 0> (DF)
09:51:51.289391 my.host.com.http > 62.165.191.132.43132: S 2149320996:2149320996(0) ack 2429552204 win 16060 <mss 1460,sackOK,timestamp
70260139 673244545,nop,wscale 0> (DF)
09:51:52.483116 62.165.191.132.43132 > my.host.com.http: . ack 1 win 5840 <nop,nop,timestamp 673244678 70260139> (DF)
09:51:52.485683 62.165.191.132.43132 > my.host.com.http: P 1:19(18) ack 1 win 5840 <nop,nop,timestamp 673244678 70260139> (DF)
09:51:52.485979 my.host.com.http > 62.165.191.132.43132: . ack 19 win 16060 <nop,nop,timestamp 70260259 673244678> (DF)
09:51:52.487368 my.host.com.http > 62.165.191.132.43132: P 1:581(580) ack 19 win 16060 <nop,nop,timestamp 70260259 673244678> (DF)
0x0000  4500 0278 4570 4000 4006 203b xxxx xxxx  E..xEp@.@.;....
0x0010  3ea5 bf84 0050 a87c 801c 0925 90d0 065e  >....P.|...%...^
0x0020  8018 3ebc 628e 0000 0101 080a 0430 1623  ..>.b.....0.#
0x0030  2820 e606 4854 5450 2f31 2e31 2034 3030  (...HTTP/1.1.400
0x0040  2042 6164 2052 6571 7565 7374 0d0a 4461  .Bad.Request..Da
0x0050  7465 3a20 5361 742c 2031 3820 4a61 6e20  te:..Sat,.18.Jan.
0x0060  3230 3033 2030 393a 3535 3a31 3020 474d  2003.09:55:10.GM
0x0070  540d 0a53 6572 7665 723a 2041 7061 6368  T..Server:..Apach
0x0080  652f 312e 332e 3236 2028 556e 6978 2920  e/1.3.26.(Unix).
0x0090  4465 6269 616e 2047 4e55 2f4c 696e 7578  Debian.GNU/Linux
0x00a0  0d0a 436f 6e6e 6563 7469 6f6e 3a20 636c  ..Connection:.cl
0x00b0  6f73 650d 0a54 7261 6e73 6665 722d 456e  ose..Transfer-En
0x00c0  636f 6469 6e67 3a20 6368 756e 6b65 640d  coding:..chunked.
0x00d0  0a43 6f6e 7465 6e74 2d54 7970 653a 2074  .Content-Type:..t
```

0x00e0	6578 742f 6874 6d6c 3b20 6368 6172 7365	ext/html;chase
0x00f0	743d 6973 6f2d 3838 3539 2d31 0d0a 0d0a	t=iso-8859-1....
0x0100	3136 630d 0a3c 2144 4f43 5459 5045 2048	16c..<!DOCTYPE.H
0x0110	544d 4c20 5055 424c 4943 2022 2d2f 2f49	TML.PUBLIC."-//I
0x0120	4554 462f 2f44 5444 2048 544d 4c20 322e	ETF//DTD.HTML.2.
0x0130	302f 2f45 4e22 3e0a 3c48 544d 4c3e 3c48	0//EN">.<HTML><H
0x0140	4541 443e 0a3c 5449 544c 453e 3430 3020	EAD>.<TITLE>400.
0x0150	4261 6420 5265 7175 6573 743c 2f54 4954	Bad.Request</TIT
0x0160	4c45 3e0a 3c2f 4845 4144 3e3c 424f 4459	LE>.</HEAD><BODY
0x0170	3e0a 3c48 313e 4261 6420 5265 7175 6573	>.<H1>Bad.Reques
0x0180	743c 2f48 313e 0a59 6f75 7220 6272 6f77	t</H1>.Your.brow
0x0190	7365 7220 7365 6e74 2061 2072 6571 7565	ser.sent.a.reque
0x01a0	7374 2074 6861 7420 7468 6973 2073 6572	st.that.this.ser
0x01b0	7665 7220 636f 756c 6420 6e6f 7420 756e	ver.could.not.un
0x01c0	6465 7273 7461 6e64 2e3c 503e 0a63 6c69	derstand.<P>.cli
0x01d0	656e 7420 7365 6e74 2048 5454 502f 312e	ent.sent.HTTP/1.
0x01e0	3120 7265 7175 6573 7420 7769 7468 6f75	1.request.withou
0x01f0	7420 686f 7374 6e61 6d65 2028 7365 6520	t.hostname.(see.
0x0200	5246 4332 3631 3620 7365 6374 696f 6e20	RFC2616.section.
0x0210	3134 2e32 3329 3a20 2f3c 503e 0a3c 4852	14.23):./<P>.<HR
0x0220	3e0a 3c41 4444 5245 5353 3e41 7061 6368	>.<ADDRESS>Apach
0x0230	652f 312e 332e 3236 2053 6572 7665 7220	e/1.3.26.Server.
0x0240	6174 2031 3932 2e31 3638 2e32 302e 3320	at.my.host.com.
0x0250	506f 7274 2038 303c 2f41 4444 5245 5353	Port.80</ADDRESS
0x0260	3e0a 3c2f 424f 4459 3e3c 2f48 544d 4c3e	>.</BODY></HTML>
0x0270	0a0d 0a30 0d0a 0d0a	...0....

```

09:51:52.487459 my.host.com.http > 62.165.191.132.43132: F 581:581(0) ack 19 win 16060 <nop,nop,timestamp 70260259 673244678> (DF)
09:51:53.667783 62.165.191.132.43132 > my.host.com.http: . ack 581 win 6960 <nop,nop,timestamp 673244797 70260259> (DF)
09:51:53.706483 62.165.191.132.43132 > my.host.com.http: . ack 582 win 6960 <nop,nop,timestamp 673244801 70260259> (DF)
09:52:11.682533 62.165.191.132.43132 > my.host.com.http: F 19:19(0) ack 582 win 6960 <nop,nop,timestamp 673246598 70260259> (DF)
09:52:11.682792 my.host.com.http > 62.165.191.132.43132: . ack 20 win 16060 <nop,nop,timestamp 70262178 673246598> (DF)

```

After the confirmation on the Apache version it now attempts to exploit this host.

```

09:51:53.669812 62.165.191.132.43178 > my.host.com.https: S 2430610185:2430610185(0) win 5840 <mss 1460,sackOK,timestamp 673244797
0,nop,wscale 0> (DF)
09:51:53.670111 my.host.com.https > 62.165.191.132.43178: S 2162823564:2162823564(0) ack 2430610186 win 16060 <mss 1460,sackOK,timestamp
70260377 673244797,nop,wscale 0> (DF)
09:51:54.250915 62.165.191.132.43178 > my.host.com.https: . ack 1 win 5840 <nop,nop,timestamp 673244846 70260377> (DF)
09:51:54.257787 my.host.com.https > 62.165.191.132.43178: P 1:40(39) ack 1 win 16060 <nop,nop,timestamp 70260436 673244846> (DF)
.....<snip>.....Banner of service running on that port returned here, which is not https.
09:51:54.319171 62.165.191.132.43178 > my.host.com.https: . ack 40 win 5840 <nop,nop,timestamp 673244862 70260436> (DF)
09:52:11.687652 62.165.191.132.43178 > my.host.com.https: R 1:1(0) ack 40 win 5840 <nop,nop,timestamp 673246598 70260436> (DF)

```

Source of Trace:

The detect came from my home network which has a dedicated screened subnet for the purposes of this project. The subnet is protected by stateful firewall based on IPtables and Linux. Traffic enters this network via Masqueraded address.

Detect Generated by:

The Detect was generated by Snort Version 1.9.0 (Build 209) with the following signature:

```

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"EXPERIMENTAL WEB-MISC bad HTTP/1.1 request, potential worm attack";
flow:to_server,established; content:"GET / HTTP/1.1|0d 0a 0d 0a|"; offset:0; depth:18;
reference:url,securityresponse.symantec.com/avcenter/security/Content/2002.09.13.html; classtype:web-application-activity; sid:1881; rev:2;)

```

The signature looks at URL get request an internal web server with the 0d 0a 0d 0a binary patter in the stream.

Probability the Source Address was Spoofed:

Low. The probability is low because the host is located behind a stateful firewall and this requires that a three-way handshake to take place. The Attacker is also interested in the response of the web server before commencing the second phase of the attack.

Attack Description:

This is two phased attack.

Phase 1, The Attacker is trying to determine what web server is running and the OS it is running on.

Phase 2, The Attacker attempts to exploit a weakness in OpenSSL SSLv2 that theApache ssl_mod uses.

Attack Mechanism:

A buffer overflow has been reported in the handling of the client key value during the negotiation of the SSLv2 protocol. A malicious client may be able to exploit this vulnerability to execute arbitrary code as the vulnerable server process, or possibly to create a denial of service condition.

What is not shown in the log is the following HTTP GET request in effort to keep the length of the document shorter:

GET / HTTP/1.1\r\n\r\n

This actually an invalid get request as it is missing a "Host:" parameter, but what we do see is the response for the invalid get request
HTTP/1.1.400 Bad.Request
Date: Sat, 18 Jan 2003 09:55:10 GMT
Server: Apache/1.3.26 (Unix) Debian.GNU/Linux
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1

The attacking host looks at the response from the from the server and in particular the "Server: " part of the response. If it is a Vulnerable version of Apache it then starts the second phase of he attack which is the mod_ssl exploit carried out on port 443. If it the exploit is successful opens shell on the victim and proceeds to upload a file in uuencode format, then uudecodes and compiles it. The code also has some peer to peer network functionality, which probably means that it can be for coordinated attacks on other systems.

Correlations:

<http://online.securityfocus.com/bid/5363/discussion/>
<http://securityresponse.symantec.com/avcenter/security/Content/2002.09.13.html>

Evidence of Active Targeting:

This host was actively targeted as this is 2 phased attack, first the reconnaissance phase to see the if the host is running a vulnerable version of Apache. Then the attack phase attempting to exploit the ssl module on apache.

Severity:

The formula used to rank the severity of the incident is as follows:
(Target Criticality + Attack Lethality)-(System Countermeasures + Network Countermeasures) = Attack Serverity

Each element is ranked 1 to 5, 1 being low, 5 being high. The maximum score (which is also the worst-case scenario) is 8. The minimum score (which is the best-case scenario) is -8.

Target Criticality: 2

This is a machine used for testing purposes.

Attack Lethality: 5

If this successful it would allow an attacker to gain root privileges remotely

System Countermeasures: 3

The Attacker was able to determine the version of the web server, however it is a different service that is listening on 443 and not apache.

Network Countermeasures: 1

The attack made it through to the host.

Attack Severity: $(2+5)-(3+1) = 3$

Defensive Recommendations:

Remove the banner from the service listening on port 443 patch the web server. This may be also prevented with the use of Intrusion detection gateway like hogwash. An application proxy firewall would not really work well here as most only provide rudimentary methods for scanning URL's. The attack actually happens on an encrypted channel, the only real way to stop the attack would be to, identify the source from the reconnaissance and block further traffic from that source address. Only problem with that is what if one host is used for reconnaissance and one for the attack ?

Multiple Choice Question:

When configuring web services for the public Internet, you should make sure that:

- a) every possible function has been installed just incase you need to use it.
- b) all sample scripts are there so that you have plenty of examples to follow when writing your own scripts.
- c) That it is an older stable version.
- d) It is the most recent secure version and all dependencies also have the latest security patches.

Answer d). When configuring web services for that internet it is imperative that you are running the most up-to-date security patches and that the services are properly configured. It is only a matter of time before a new worm is written or someone finds your server, and that service should be secure as possible.

Assignment-3 [\(To Top of Document\)](#)

Analyze This !

Executive Summary [\(To Top of Document\)](#)

The TCP/IP protocol suite is basis for all communications used by Hosts to communicate with one another on the Internet. This Suite of Protocols and computer applications that use it can have flaws in the way they have been programmed that may allow someone with malicious intent to take advantage of. Since the majority of businesses rely on computer systems these days it is the best interests of the business to protect these assets, hence the need for Intrusion Detection Systems. This is the same as a business installing a burglar alarm and locks on their doors to protect their physical assets.

This is an analysis of log files generated by an Open Source Network Intrusion Detection System (NIDS) called Snort. Snort has been configured to send log output to a Unix system Process called Syslog for all intrusion alerts. Snort also logs other types of activity known as port scans to an ASCII text file, this type of behavior is usually associated with reconnaissance from a potential attacker and can shed some light on the potential attackers disposition. Finally we have a third type of log file for what is colloquially known as Out Of Specification Packets (OOS), these logs represent abnormal use of the TCP/IP protocol suite, just like someone not obeying the road traffic laws.

During this investigation it is apparent that the Snort configuration, Snort rules, host defenses, perimeter defenses and network design are sub optimal. The Snort configuration appears that it lacks definitions of what is the Internal network, which are the hosts that you are trying to protect and what is the External network, the hosts that you are trying to defend against. This also impacts the signatures used. If no clear definitions are made for what are the internal and external networks, it often results in high number of false positives.

Network defenses and design also have an impact on the amount of alerts generated, In the course of the report it is evident that there is a very loose or no perimeter defense in place as traffic is able to traverse from external sources directly into the internal network. If this is allowed the hosts that have exposure need to be harden against public misuse. Also host that are meant to provide public services should be located on screened DMZ so that they can't be used as a staging point into the internal network.

We also see that there is an unacceptable use of network resources with the presence of eDonkey, Kazaa , Napster and chat programs being used. This does not only consume network resources but also allows programs from dubious sources to be downloaded and executed. This seriously hampers efforts in trying to lock down network security.

The rest of the report will attempt to point out the hosts that may be compromised and require further investigation. An attempt will also be made to demonstrate the relationships between some of the internal hosts and illustrates the statements above about the general condition of the network. Defensive recommendations are mentioned with each type of event and how it could have helped to protect the internal network and reduce the amount of alerts or activity observed.

This report also assumes the reader has a general working knowledge of network components and the TCP/IP Protocol Suite.

Next is an output from one of the log entries and an explanation of the different fields, as this may be helpful to the understanding of the reader:

04/01-13:16:50.757401 **[**]** connect to 515 from inside **[**]** MY.NET.150.83:527 -> MY.NET.151.77:515

04/01-13:16:50.757401 is a time date stamp of when the event happened, 04 being march, 01 being the first day of march, followed by the time of day in millionths of a second at the systems clock.

[] connect to 515 from inside [**]** is the actual event.

MY.NET.150.83:527 is the source address separated by a ":" and then the source port

-> is the direction indicated as an arrow

MY.NET.151.77:515 is the destination address separated by ":" and then the destination port.

Summary of Alerts [\(To Top of Document\)](#)

Total Alerts : 1135976 + 70 (including Out of Spec and excluding preprocessor port scans in alert logs)
Time Span : 03/31 00:00:17 -> 04/08 23:57:19
Out of Spec : 04/01 00:54:21 -> 04/08 17:43:17

Total Scans : 5421875
Time Span : 03/31 00:00:01 -> 04/09 00:00:03

Summary of Events

Count	Events
637379	connect to 515 from inside
100061	spp_http_decod IIS Unicode attack detected
96461	SNMP public access
90017	SMB Name Wildcard
44877	ICMP Echo Request L3retriever Ping

35998 spp_http_decod CGI Null Byte attack detected
35659 MISC Large UDP Packet
23101 INFO MSN IM Chat data
16897 High port 65535 udp - possible Red Worm - traffic
15262 INFO Inbound GNUTella Connect request
7285 ICMP Echo Request Nmap or HPING2
5748 Watchlist 000220 IL-ISDNNET-990517
4945 FTP DoS ftpd globbing
4830 ICMP Fragment Reassembly Time Exceeded
3506 INFO Outbound GNUTella Connect request
3163 Incomplete Packet Fragments Discarded
2114 WEB-IIS view source via translate header
1748 ICMP Router Selection
1264 WEB-MISC Attempt to execute cmd
843 NMAP TCP ping!
659 Watchlist 000222 NET-NCFC
473 WEB-IIS _vti_inf access
444 WEB-FRONTPAGE _vti_rpc access
350 Null scan!
291 ICMP Echo Request Windows
271 INFO napster login
227 Possible trojan server activity
225 WEB-CGI scriptalias access
203 SCAN Proxy attempt
183 INFO FTP anonymous FTP
169 ICMP Destination Unreachable (Communication Administratively Prohibited)
109 IDS552/web-iis_IIS ISAPI Overflow ida nosize
107 INFO Possible IRC Access
101 ICMP traceroute
88 WEB-CGI rsh access
74 WEB-CGI ksh access
73 INFO Napster Client Data
63 WEB-MISC 403 Forbidden
58 INFO - Possible Squid Scan
54 FTP CWD / - possible warez site
43 WEB-MISC compaq nsight directory traversal

40 EXPLOIT x86 NOOP
38 EXPLOIT NTPDX buffer overflow
36 Queso fingerprint
36 Attempted Sun RPC high port access
35 MISC traceroute
33 SCAN Synscan Portscan ID 19104
31 EXPLOIT x86 setuid 0
28 Russia Dynamo - SANS Flash 28-jul-00
26 ICMP Destination Unreachable (Protocol Unreachable)
23 SUNRPC highport access!
21 Port 55850 tcp - Possible myserver activity - ref. 010313-1
20 Back Orifice
16 High port 65535 tcp - possible Red Worm - traffic
13 WEB-MISC http directory traversal
12 ICMP Echo Request BSDtype
11 RPC tcp traffic contains bin_sh
11 EXPLOIT x86 setgid 0
9 Tiny Fragments - Possible Hostile Activity
9 SCAN FIN
9 MYPARTY - Possible My Party infection
8 Port 55850 udp - Possible myserver activity - ref. 010313-1
7 EXPLOIT x86 stealth noop
5 WEB-MISC whisker head
5 WEB-MISC prefix-get //
5 WEB-IIS Unauthorized IP Access Attempt
5 SYN-FIN scan!
4 WEB-MISC ICQ Webfront HTTP DOS
4 TFTP - External UDP connection to internal tftp server
3 WEB-CGI formmail access
3 TFTP - Internal TCP connection to external tftp server
3 MISC PCAnywhere Startup
3 INFO Outbound GNUTella Connect accept
3 INFO Inbound GNUTella Connect accept
2 x86 NOOP - unicode BUFFER OVERFLOW ATTACK
2 suspicious host traffic
2 WEB-MISC webdav search access

2	WEB-IIS encoding access
2	TFTP - Internal UDP connection to external tftp server
2	RFB - Possible WinVNC - 010708-1
2	Probable NMAP fingerprint attempt
2	EXPLOIT x86 NOPS
1	WEB-IIS asp-dot attempt
1	WEB-CGI redirect access
1	TELNET access
1	TCP SMTP Source Port traffic
1	MISC source port 53 to <1024
1	INFO napster upload request
1	INFO - Web Dir listing
1	IDS475/web-iis_web-webdav-propfind
1	ICMP Destination Unreachable (Host Unreachable)

Connect to 515 from inside

This Alert is generated from a snort signature where the third part of the TCP three way hand shake has been completed and looks for the “ACK” flags being set for a service that connects to port 515. During a normal TCP session the “ACK” flag can be seen multiple times for a single session especially when there is a large amount of data going flowing in that session. Port 515 is a TCP-based protocol for which the line printer daemon listens. The data below explains why Alert is seen.

Top 10 Destination Addresses

count	destip
317953	MY.NET.151.77
314877	MY.NET.150.198
4470	MY.NET.150.83
79	MY.NET.1.63

Top 10 Source Addresses

count	souceip
317953	MY.NET.150.83
56133	MY.NET.153.118

28328	MY.NET.153.126
23126	MY.NET.153.119
19108	MY.NET.153.164
9459	MY.NET.153.136
8528	MY.NET.153.113
8356	MY.NET.153.211
7500	MY.NET.153.123
6733	MY.NET.153.121

Conducting an investigation on the Top 10 source addresses can lead to allows us to draw conclusions on the nature of this alert

We can see that the all the Top 10 source addresses come from the within the clients internal network. MY.NET.150.83 is the number one Source address as it generate the most amount of these alerts being a total of 317953. Below is an excerpt from the Alert logs.

```
04/01-13:16:50.757401 [**] connect to 515 from inside [**] MY.NET.150.83:527 -> MY.NET.151.77:515
04/01-13:16:50.757479 [**] connect to 515 from inside [**] MY.NET.150.83:527 -> MY.NET.151.77:515
04/01-13:16:51.264435 [**] connect to 515 from inside [**] MY.NET.150.83:527 -> MY.NET.151.77:515
04/01-13:16:51.264500 [**] connect to 515 from inside [**] MY.NET.150.83:527 -> MY.NET.151.77:515
04/01-13:16:55.274557 [**] connect to 515 from inside [**] MY.NET.150.83:528 -> MY.NET.151.77:515
04/01-13:16:55.275101 [**] connect to 515 from inside [**] MY.NET.150.83:528 -> MY.NET.151.77:515
```

After running a query on these logs looking for unique event “connect to 515 from inside” with the unique source address of “MY.NET.150.83”, we see that all events are to the destination host “MY.NET.151.77”.

souceip	sourceport	destip	destport
MY.NET.150.83	512	MY.NET.151.77	515
MY.NET.150.83	513	MY.NET.151.77	515
MY.NET.150.83	514	MY.NET.151.77	515

[RFC1179](#) states that “the source port must be in the range 721 to 731, inclusive” for the Line Printer Daemon (LPD). This is not the case for the source address “MY.NET.150.83”, all the source ports range from 512 to 564. It is interesting to note that most of these source ports are using the “r” services for unix systems, 512 being the Remote execution service, 513 rlogin and 514 being rcmd. Below is sample of traffic taken from a “clean” network using tcpdump demonstrating normal LPD use.

```

MY.CLEANSOURCE.0.144.727 > MY.CLEANDEST.60.96.515: S 496043440:496043440(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
MY.CLEANDEST.60.96.515 > MY.CLEANSOURCE.0.144.727: S 1627762414:1627762414(0) ack 496043441 win 17520 <mss
1460,nop,nop,sackOK> (DF)
MY.CLEANSOURCE.0.144.727 > MY.CLEANDEST.60.96.515: . ack 1 win 17520 (DF)
MY.CLEANSOURCE.0.144.727 > MY.CLEANDEST.60.96.515: P 1:11(10) ack 1 win 17520 (DF)
MY.CLEANDEST.60.96.515 > MY.CLEANSOURCE.0.144.727: P 1:2(1) ack 11 win 17510 (DF)
MY.CLEANDEST.60.96.515 > MY.CLEANSOURCE.0.144.727: F 2:2(0) ack 11 win 17510 (DF)
MY.CLEANSOURCE.0.144.727 > MY.CLEANDEST.60.96.515: F 11:11(0) ack 2 win 17519 (DF)
MY.CLEANSOURCE.0.144.727 > MY.CLEANDEST.60.96.515: . ack 3 win 17519 (DF)
MY.CLEANDEST.60.96.515 > MY.CLEANSOURCE.0.144.727: . ack 12 win 17510 (DF)

```

The second highest in the Top 10 Sources is “MY.NET.153.118” also does not conform to [RFC1179](#), however all the source ports are in the ephemeral port range, that is source ports are above 1023. This connection is also only between the two hosts “MY.NET.153.118” and “MY.NET.150.198”. This maybe an incorrect implementation of the LPD service.

```

03/31-12:56:17.555598 [**] connect to 515 from inside [**] MY.NET.153.118:1299 -> MY.NET.150.198:515
03/31-12:56:17.556154 [**] connect to 515 from inside [**] MY.NET.153.118:1299 -> MY.NET.150.198:515
03/31-12:56:17.556222 [**] connect to 515 from inside [**] MY.NET.153.118:1299 -> MY.NET.150.198:515

```

The Same behavior is observed for MY.NET.153.126, MY.NET.153.119, MY.NET.153.164, MY.NET.153.136, MY.NET.153.113, MY.NET.153.211 and MY.NET.153.121 and all go to the same destination “MY.NET.150.198”

Recommendations:

Although there have been documented worms that make use of the LPD daemon to propagate themselves such as the [Ramon Worm](#) and [lpdw0rm](#) the behavior of the source addresses do not exhibit the expected behavior of a worm, scanning multiple systems for a certain port and then propagating itself, all the traffic is seen from the top 10 sources is only to two destinations which may indicate a printing relationship. It would be recommended to check these systems to see if this is indeed the case.

Secondly check the destination systems for vulnerable versions of the LPD daemon and patch them, vulnerabilities for the LPD daemon are outline on the following web sites <http://www.sans.org/newlook/alerts/port515.htm> and the neohapsis archives [CAN-2000-0839](#). It would also recommend that the source “MY.NET.150.83” be either put on a watch list and/or investigated more closely as it make use of privileged ports, in particular the “i” services. Once this investigation is completed the sensor policies are tuned to take into account of what acceptable traffic is allowed on the internal Network.

spp_http_decode IIS Unicode attack detected

This is an Alert caused by the Snort Http_decode pre-processor. In the alert summary you see it as spp_http_decode as the scripts that passed the log files remove the “.” and the character immediately preceding it.

In version 4 and 5 of IIS users may be able to access any file located on the same logical drive as the web server directory by entering a "/" or "\" or "../". The file can be executed, added or deleted with the same rights of the IIS service. This was a vulnerability exploited by the Code Blue Worm.

More information can be found here <http://online.securityfocus.com/bid/1806/discussion/> or here

http://www.sans.org/newlook/resources/IDFAQ/iis_unicode.htm.

This snort pre-processor or at least the older versions generate large amounts of false positives with this feature enabled. With out the payload it is difficult to tell what really happened.

Due to the volume it of alerts it could be a worm, however the logs don't really indicate scanning activity of an infected host looking for other machines.

Now there is another reason why we see use of Unicode characters. Asian languages use double byte characters to represent the Asian written characters.

APNIC is the governing body for IP allocations in the Asia pacific region.

If you look at <http://www.apnic.net/db/ranges.html> you can see the IP v4 allocation to APNIC is as follows:

61.0.0.0/8, 202.0.0.0/7, 210.0.0.0/7, 218.0.0.0/7 220.0.0.0/7

If we look at all destination address with the destination port 80 that are outside the internal network for this alert we see that a lot of these address are from Asia Pacific.

```
$ cat alert.020??? | grep "IIS Unicode attack detected" > unicode.txt
```

```
$ cat unicode.txt | cut -d ">" -f 2 | grep -v MY | egrep -e "80$" | sed -e 's/^ //' | cut -d "." -f -3 | sort -u > temp.txt
```

```
$ cat temp.txt
```

```
--<snip>--
```

```
202.1.232
```

```
202.1.237
```

```
202.1.238
```

```
--<snip>--
```

```
203.254.192
```

```
203.255.3
```

```
204.253.104
```

```
--<snip>--
```

```
209.225.0
```

```
210.106.213
```

```
210.112.177
```

```
--<snip>--
```

```
211.78.38
```

```
211.99.211
```

```
212.187.205
```

```
--<snip>--
```

And looking at some of these IP spaces below, we see that they indeed come from Asia Pacific, and even China.

```
[~]# whois -h whois.apnic.net 202.1.232.0
```



```
% [whois.apnic.net node-2]
% How to use this server    http://www.apnic.net/db/
% Whois data copyright terms  http://www.apnic.net/db/dbcopyright.html
```

```
inetnum:    202.1.232.0 - 202.1.239.255
netname:    YAHOO-ASIA
descr:      streaming media, e-mail, instant messenger, www,  etc
```

```
[~]# whois -h whois.apnic.net 202.96.170.0
% [whois.apnic.net node-2]
% How to use this server    http://www.apnic.net/db/
% Whois data copyright terms  http://www.apnic.net/db/dbcopyright.html
```

```
inetnum:    202.96.128.0 - 202.96.191.255
netname:    CHINANET-GD
descr:      CHINANET Guangdong province network
descr:      Data Communication Division
descr:      China Telecom
```

Recommendations: Turn on the TCPDump Output plug for snort and investigate the payload further. Look at what is how the internal and external network is defined in the Snort.conf file so that this only trigger on inbound connections to internal web servers. Also when analyzing this data, it would be wise to correlate events from the web server logs like error 403's, which may suddenly give highlight machines that have made previous unsuccessful attacks.

SNMP public access

Many companies employ the use of SNMP for network management using it to get status and health checks of network hosts and devices, below is the statistics for this alert below.

Top 10 Source Addresses

count	souceip
18902	MY.NET.70.177
8869	MY.NET.88.203
8843	MY.NET.88.181
8832	MY.NET.88.207

8776	MY.NET.88.145
8755	MY.NET.88.159
8148	MY.NET.88.136
7017	MY.NET.150.198
4242	MY.NET.88.251
3239	MY.NET.153.220

Without knowing the details of this signature and how the client has setup the snort configuration it is difficult to draw any conclusions about this alert. While there have been new vulnerabilities have been announced in mid 2002 and also documented use of [SNMP for reconnaissance](#). We see that all the source and destinations are within the client's network as stated before SNMP is used for Network management and if it is being used for this purpose it would generate a large amount of false positives. The Source and source addresses do not generate suspicious behavior for this alert

```
03/31-00:36:25.762785 [**] SNMP public access [**] MY.NET.150.198:1661 -> MY.NET.87.215:161
```

```
03/31-00:36:25.770423 [**] SNMP public access [**] MY.NET.150.198:1662 -> MY.NET.87.215:161
```

```
03/31-00:36:25.775496 [**] SNMP public access [**] MY.NET.150.198:1663 -> MY.NET.87.215:161
```

If we "grep" through the logs looking for instance of this alert where an outside source was involved it may give us concern for immediate alarm. If there is no activity from an outside source it does not necessarily mean that exploit has not occurred but it gives us more of a sanity check that as if is a problem in the signature that caused this alert, unless SNMP is not supposed to be used on the internal network of course.

Checking alert logs for instances of external addresses in source addresses:

```
$ cat alert.020??? | grep "SNMP public access" | cut -d " " -f 8 | grep -v MY.NET
```

No Results returned

Checking alert logs for instances of external addresses in destination addresses:

```
$ cat alert.020??? | grep "SNMP public access" | cut -d " " -f 10 | grep -v MY.NET
```

No Results returned

There seems to be no indication of scanning activity, but more like the activity you would expect to see from a network management station.

Activity for this type of Alert has been seen in the wild and can be referenced at:

<http://www.sans.org/y2k/051200.htm>

http://ki.sei.cmu.edu/idar/drill_attack.cfm?attack=SNMP%20Grabbing

The signature may have looked like this taken from <http://scorpions.net/~fygrave/snort/misc-lib>

```
alert udp any any -> $HOME_NET 161 (msg: "SNMP public access"; content:"public";)
```

If the signature did look like this it could explain the high volume of events seen for "SNMP public access" which are more than likely to be false positives.

Recommendations: Refine the signature so that it reduces the amount of false positives. Start thinking about upgrading all the SNMP agents to version 3. Also change the public community string to something else and make sure that there is adequate perimeter defense so that it reduces the chances of a compromise.

SMB Name Wildcard

Prelude: SMB wildcards alerts relate to the NetBios name service. This service commonly used by Windows machines using a file and print sharing protocol to obtain domain, user and host information, Unix machines can also make use of this protocol for the SAMBA service.

Top 10 Source Addresses

count	souceip
20649	MY.NET.11.6
15857	MY.NET.11.7
6696	MY.NET.11.5
1114	MY.NET.152.177
1082	MY.NET.152.161
1025	MY.NET.152.168
990	MY.NET.152.167
956	MY.NET.152.166
928	MY.NET.152.21
913	MY.NET.152.171
Total number of Rows: 10	

An attacker can make use of this protocol to gather reconnaissance information about a target host to launch a future attack. This traffic can be generated by legitimately when browsing for a computer and the IP address is only known (ie in Windows "Start->Search->find computer") and is demonstrated in the preceeding trace using "tcpdump -X".

```
09:33:13.411613 MY.NET.89.172.1666 > MY.NET.89.171.139: S 1125371993:1125371993(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
09:33:13.411613 MY.NET.89.171.139 > MY.NET.89.172.1666: R 0:0(0) ack 1 win 0 (DF)
09:33:13.411613 MY.NET.89.172.1665 > MY.NET.89.171.445: S 1125306921:1125306921(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
09:33:13.411613 MY.NET.89.171.445 > MY.NET.89.172.1665: R 0:0(0) ack 1 win 0 (DF)
09:33:13.911613 MY.NET.89.172.1666 > MY.NET.89.171.139: S 1125371993:1125371993(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
09:33:13.911613 MY.NET.89.171.139 > MY.NET.89.172.1666: R 0:0(0) ack 1 win 0 (DF)
09:33:13.911613 MY.NET.89.172.1665 > MY.NET.89.171.445: S 1125306921:1125306921(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
09:33:13.911613 MY.NET.89.171.445 > MY.NET.89.172.1665: R 0:0(0) ack 1 win 0 (DF)
09:33:13.911613 MY.NET.89.172.137 > MY.NET.89.171.137:
>>> NBT UDP PACKET(137): QUERY; REQUEST; UNICAST
0x0000 4500 004e 2440 0000 8011 0ae7 ac10 59ac      E..N$@.....Y.
0x0010 ac10 59ab 0089 0089 003a 3294 8222 0000      ..Y.....:2.."..
```

0x0020 0001 0000 0000 0000 2043 4b41 4141 4141CKAAAAA
0x0030 4141 4141 4141 AAAAAA

The signature probably looked similar to this one obtained from Jeff Holland's GCIA assignment,

```
alert UDP $EXTERNAL any -> $INTERNAL 137 (msg: "IDS177/netbios_netbios-name-query"; content:
"CKAAAAAAAAAAAAAAAAAAAAAAAAAAAA[00 00]";)
```

And is documented in the arachNIDS database, located at <http://www.whitehats.com/IDS/177>,

"Additionally, some desktop firewalls will automatically send the packets to any other host that connects back to the user (such as identd requests from a mail server or IRC server). NetBIOS name traffic is considered background noise on the network and should only be considered when combined with other forensic evidence that points to a problem/suspicion." [12].

The majority of the traffic is between MY.NET.11.5, MY.NET.11.6, MY.NET.11.7 or to other hosts on the internal network. These events do not resemble any scanning activity and I would think that this signature is prone to false positives especially if the variable for \$EXTERNAL does not define external networks correctly. This seems to be the case in this instance as the alert is generated from traffic amongst internal hosts.

Recommendations: Microsoft actually recommends when running a Microsoft network to have a strong perimeter defense, so this should be implemented. Also this signature should be revised that it actually reflects the correct definition for external networks.

ICMP Echo Request L3retriever Ping

Statistics for Event: "ICMP Echo Request L3retriever Ping"

Top 3 Destination Addresses

count	destip
20743	MY.NET.11.6
15874	MY.NET.11.7
6750	MY.NET.11.5

Top 10 Source Addresses

count	souceip
1116	MY.NET.152.177
1078	MY.NET.152.161

1031	MY.NET.152.168
992	MY.NET.152.167
975	MY.NET.152.166
946	MY.NET.152.21
919	MY.NET.152.171
881	MY.NET.152.163
866	MY.NET.152.162
858	MY.NET.152.157

Total number of Rows: 10

According to the [Whitehats](#) database the signature can be generated by a legitimate security scanning tool called “Retriever 1.5” which should only be used during an authorized security audit or a false positives from Win2K host talking to Win2K domain controllers. Looking closer at the at the top 10 source addresses generating this alert they are generating traffic to the following three internal hosts. The 3 internal destination hosts are also the first 3 members of the Top 10 destinations for this alert.

MY.NET.11.7

MY.NET.11.5

MY.NET.11.6

If these are 3 destination hosts are domain controllers for the customer’s network or workstations conducting authorized network audits then this would be a false positive.

Recommendations: The sensor should be tuned to take these domain controllers or work stations into account. Further more check the other 7 of the Top 10 destinations to see if they are also domain controllers. If the is not the conduct a further investigation into the source generating this alert.

spp_http_decode CGI Null Byte attack detected

This alert is generated by the snort http decode pre-processor. Due to the processing scripts we see the misspelling of spp_http_decode

If the Snort pre-processor sees a %00 in the http request it will generate alert. You may see sites that use cookies with urlencoded binary data that may cause these false positives. Unless you have a packet capture you can never be sure. More information on this alert can be found here

http://www.sans.org/newlook/resources/IDFAQ/iis_unicode.htm

All External traffic seems to going to MY.NET.5.96, the other alerts are from internal host going to external web servers without any real signs of suspicious behavior.

Recommendations: Check that the MY.NET.5.96 is supposed to be a web server that is accessible from the public if not, implement effective perimeter defense so that no public connections can be made to it. Check that this is also running the recommend security patches for that operating system and web server. It may also be advisable to use the TCPDump output plug in to verify the payload of the packet causing the alert.

MISC Large UDP Packet

Below is an extract from the logs with this alert

```
03/31-16:22:42.185444  [**] MISC Large UDP Packet [**] 211.233.70.163:2080 -> MY.NET.153.106:3494
03/31-16:22:42.283179  [**] MISC Large UDP Packet [**] 211.233.70.163:2080 -> MY.NET.153.106:3494
03/31-16:22:42.378680  [**] MISC Large UDP Packet [**] 211.233.70.163:2080 -> MY.NET.153.106:3494
```

Statistics for Event: "MISC Large UDP Packet"

Top 10 Destination Ports

count	destport
-------	----------

7261	1326
6078	1769
3977	1791
1430	1647
1406	2125
1106	2407
1038	1709
941	2573
922	1633
893	1672

Total number of Rows: 10

These are the following UDP port register by IANA

- 1326 – WinSic
- 1769 – Autocad software
- 1791 – eContent web based content management system
- 1647 – RSAP used for routing in networks using Network Address translation
- 2125 – Lockstep security Product for repairing web sites when Hacked
- 2407 – Multiplexing software
- 1709 – Centra Video conferenceing software
- 2573 - Video Conferenecing software by IBM

If we do a whois on all the Top 10 source addresses then we can see that most of them come from either Korea Telecom or China-Net except for 63.240.15.205 and the 216.106.173.x address which is registered to ATT networks and Ibeam respectively.

```
inetnum: 61.78.32.0 - 61.78.35.255
netname: KORNET-IDC-JUNGANG-KTIDC-KR
descr: CENTRAL DATA COMMUNICATION OFFICE
descr: 128-9 YEUNKEONDONG JONGROKU
descr: SEOUL
descr: 110-460
country: KR
.....<snip>.....
person: Won Kang
descr: KOREA TELECOM
descr: 128-9 Youngundong Chongroku
descr: SEOUL
descr: 110-460
country: KR
phone: +82-2-747-9213
fax-no: +82-2-766-5901
e-mail: ip@ns.kornet.net
nic-hdl: WK560-KR
mnt-by: MNT-KRNIC-AP
remarks: This information has been partially mirrored by APNIC from
remarks: KRNIC. To obtain more specific information, please use the
remarks: KRNIC whois server at whois.krnic.net.
changed: hostmaster@nic.or.kr 20020923
source: KRNIC
```

What to make from the traffic "MISC Large UDP Packet". One thing we don't know and that is how large are these UDP packets

We see the destinations do not appear to be going to services that are well know or implemented very often.

We see that all the top 10 sources are outside the clients network and mostly being from China-Net and Korea Telecom. If we look at the "Out of Spec" packet logs, none of the source addresses correlate with any of the Top 10 source addresses for this attack. See list below.

Source address for Out of Spec packets:

```
142.51.44.123
192.115.135.8
.....<snip>.....
68.66.64.71
```

What we can take a guess that if this is not legitimate traffic for which I have the feeling it is probably not. And it being UDP traffic going to port that has a high chance of not being used, which would elicit a response of either ICMP destination unreachable or port unreachable. It could also be some form of covert channel. I would probably take my bets that this is network mapping effort from various independent sources.

Recommendations: Check with the system administrators that there is no valid service that should be used from these sources, if not then implement some perimeter defense in the form of a stateful firewall and limit the services available from the outside world.

INFO MSN IM Chat data

INFO Inbound GNUTella Connect request

INFO Outbound GNUTella Connect request

INFO napster login

INFO Possible IRC Access

INFO Napster Client Data

WEB-MISC ICQ Webfront HTTP DOS

INFO Outbound GNUTella Connect accept

INFO Inbound GNUTella Connect accept

INFO napster upload request

INFO - Web Dir listing

All of the Above alerts are file are associated with internet file sharing and chat programs. These should not be present in a network that is concerned about security as they are open for any body to download and upload files to and from the internet with Dubious origins. As you can see these alerts are informational indicated by the INFO at the beginning of the line.

Recommendations: Implement a firewall with a tight policy and get rid of this traffic so that this type of traffic of this type is permitted across the network perimeter. Re-educate the users as to what is acceptable use of network resources

High port 65535 udp - possible Red Worm – traffic & High port 65535 tcp - possible Red Worm - traffic

The Adore Worm that was originally called the Red Worm would normally infect and propagate itself by exploiting a vulnerability in LPRng, rpc-statd, wu-ftpd and BIND and IIS. This alert is not an alert generated by one of these exploits but a back door that was left behind after the compromised by the Worm.

More information can be found at these locations:

<http://www.sans.org/y2k/ramen.htm>

http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm

<http://www.sans.org/y2k/adore.htm>

It Could have been generated a rule such as the following, but this is only a guess

alert udp any any <> any 65535 (msg:"High port 65535 udp - possible Red Worm - traffic";)

If it was a signature similar to the one above it would be prone to false positives, below you can see that the traffic possibly from the use of the Quick Mail Transfer Protocol QMTP although I have never seen this application implemented in any environment that I have worked in.

```
MY.NET.6.48:209 -> MY.NET.153.216:65535
MY.NET.6.48:209 -> MY.NET.153.216:65535
MY.NET.6.48:65535 -> MY.NET.153.216:31743
.....<snip>.....
MY.NET.6.48:209 -> MY.NET.153.216:65535
MY.NET.6.48:16637 -> MY.NET.153.169:65535
MY.NET.6.48:65535 -> MY.NET.153.176:65535
MY.NET.6.48:65535 -> MY.NET.153.176:65408
```

4145 events came from source MY.NET.6.48, 3627 from MY.NET.6.49 and 2873 from MY.NET.6.50, which seem to be the source with the majority of the events for this alert. The first thing that comes to mind is this is some sort of trace route, probably not because most of the destinations are in the one vary all in one IP block, More consistent with worm behavior.

Correlations: This event was seen in http://www.giac.org/practicles/Todd_Chapman_GCIA.doc.

Recommendations: Implement some network Auditing stations, that can scan for these type of Worm infections and also what services are being run on other workstations. These "unwanted" services can then be removed and/or patched. This also can reduce the effect on the Network when a new worm is released so that machines are only running the necessary services. Effective Border protection is also worthwhile so that at least there are choke points into the network and outbreaks can be isolated.

ICMP Echo Request Nmap or HPING2

Statics for Event: "ICMP Echo Request Nmap or HPING2"

Top 10 Destination Addresses

count	destip
3822	MY.NET.11.6
3144	MY.NET.11.7
6	MY.NET.1.3
5	209.53.113.23
2	MY.NET.88.137
2	MY.NET.88.148
2	MY.NET.88.226
2	MY.NET.88.229

2 MY.NET.88.239
2 MY.NET.150.2
Total number of Rows: 10

Top 10 Source Addresses

count	souceip
308	MY.NET.253.10
159	MY.NET.152.171
157	MY.NET.152.166
157	MY.NET.152.172
151	MY.NET.152.21
148	MY.NET.152.177
147	MY.NET.152.163
141	MY.NET.152.162
141	MY.NET.152.158
140	MY.NET.152.157

Total number of Rows: 10

This type of event is usually is a sign of some reconnaissance effort using ICMP on your network, I have seen this with some network monitoring systems based on BSD one of the networks we monitor. In our case we see alert coming from the same network management systems all the time. More information is available at <http://project.honeynet.org/scans/scan17/som/som3/IDS152.pdf>. Having a closer look at the Top 10 source addresses we can see that number one on the list MY.NET.253.10 is very suspicious. It is either an authorized network auditing station or it has been compromised and is involved in doing further reconnaissance of this network.

souceip	sourceport	destip	destport
MY.NET.253.10		MY.NET.5.79	
MY.NET.253.10		MY.NET.5.83	
MY.NET.253.10		MY.NET.153.211	
MY.NET.253.10		MY.NET.153.219	
MY.NET.253.10		MY.NET.153.220	

Total number of Rows: 299

Recommendations: Check with the systems administrator what this machine is used for. If it is not for network scanning you have a problem and better take out your book on computer forensics and start investigating this host. Also check the MY.NET.11.6 and MY.NET.11.7, but the traffic from these hosts looks a lot more benign and looks more inline with network monitoring stations.

Watchlist 000220 IL-ISDNNET-990517

Watch lists are lists of hosts where suspicious activity has been observed and have flagged for further observation.

The list below is a list of IP Addresses from this Watch list.

212.179.112.100, 212.179.125.254, 212.179.125.79, 212.179.126.3, 212.179.127.26, 212.179.127.32, 212.179.26.4,
212.179.27.176, 212.179.32.109, 212.179.35.118, 212.179.35.119, 212.179.35.121, 212.179.35.8, 212.179.35.97, 212.179.37.10,
212.179.38.163, 212.179.38.233, 212.179.38.75, 212.179.38.83, 212.179.40.132, 212.179.43.98, 212.179.44.99, 212.179.45.195,
212.179.45.208, 212.179.112.100, 212.179.125.254, 212.179.125.79, 212.179.126.3, 212.179.127.26, 212.179.127.32, 212.179.26.4,
212.179.27.176, 212.179.32.109, 212.179.35.118, 212.179.35.119, 212.179.35.121, 212.179.35.8, 212.179.35.97, 212.179.37.10, 212.179.38.163,
212.179.38.233, 212.179.38.75, 212.179.38.83, 212.179.40.132, 212.179.43.98, 212.179.44.99, 212.179.45.195, 212.179.45.208

These addresses all are from a blocks that are assigned in Israel:

```
$ whois -h whois.ripe.net 212.179.112.100
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripenc/pdb-services/db/copyright.html
```

```
inetnum: 212.179.100.0 - 212.179.124.255
netname: CABLES-CONNECTION
mnt-by: INET-MGR
descr: CABLES-CUSTOMERS-CONNECTION
country: IL
admin-c: MR916-RIPE
tech-c: ZV140-RIPE
status: ASSIGNED PA
remarks: please send ABUSE complains to abuse@bezeqint.net
remarks: INFRA-AW
notify: hostmaster@bezeqint.net
changed: hostmaster@bezeqint.net 20021029
source: RIPE
```

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

person: Miri Roaky
address: bezeq-international
address: 40 hashacham
address: petach tikva 49170 Israel
phone: +972 1 800800110
fax-no: +972 3 9203033
e-mail: hostmaster@bezeqint.net
nic-hdl: MR916-RIPE
changed: hostmaster@bezeqint.net 20021027
source: RIPE
.....<snip>.....

Looking at the sources in this watch list we can only see one other event different event generated from the this alert.

```
$ WATCHLIST=`cat alert.020??? | grep "Watchlist 000220 IL-ISDNNET-990517" | cut -d "]" -f 3- | sed -e 's:/ /g' | awk '{print $1}' | sort -u`  
$ echo $WATCHLIST  
212.179.112.100 212.179.125.254 212.179.125.79 212.179.126.3 212.179.127.26 212.  
179.127.32 212.179.26.4 212.179.27.176 212.179.32.109 212.179.35.118 212.179.35.  
119 212.179.35.121 212.179.35.8 212.179.35.97 212.179.37.10 212.179.38.163 212.1  
79.38.233 212.179.38.75 212.179.38.83 212.179.40.132 212.179.43.98 212.179.44.99  
212.179.45.195 212.179.45.208 212.179.47.79 212.179.5.90 212.179.72.6 212.179.7  
5.140  
$ for IP in $WATCHLIST ; do cat alert.020??? | grep $IP | grep -v Watchlist >> watch.txt ; done  
$ cat watch.txt  
04/08-11:53:07.320257 [**] INFO Inbound GNUTella Connect request [**] 212.179.126.3:24274 -> MY.NET.153.141:6346
```

Most of the events are generated by the use of Kazaa, which is a peer to peer file sharing application as you can see by the destination port of 1214 from the log extract below.

```
03/31-13:35:55.732628 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.125.254:1359 -> MY.NET.150.133:1214  
03/31-13:35:56.433289 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.125.254:1359 -> MY.NET.150.133:1214
```

03/31-13:35:57.134043 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.125.254:1359 -> MY.NET.150.133:1214

Recommendations: While watch list serve as useful tool for the intrusion analyst, you can often be inundated with information. You then must prioritize the alerts that you do see for investigation. It would be beneficial to tune the rule for this watch list or implement effective perimeter defense so that this type of traffic does not make it into the internal network as Kazaa is perfect mechanism to deliver unwanted Trojan's and Viruses.

FTP DoS ftpd globbing

Below is a sample out put from the logs:

```
03/31-20:49:00.483364 [**] FTP DoS ftpd globbing [**] 160.39.194.140:2522 -> MY.NET.150.46:21
03/31-20:49:00.523303 [**] FTP DoS ftpd globbing [**] 160.39.194.140:2522 -> MY.NET.150.46:21
03/31-20:49:00.583988 [**] FTP DoS ftpd globbing [**] 160.39.194.140:2522 -> MY.NET.150.46:21
```

Top 10 Source Addresses

count	souceip
1213	169.232.80.45
979	131.212.80.139
705	164.76.174.31
195	129.74.140.176
193	128.12.57.36
137	155.69.8.243
132	129.237.88.160
96	164.76.178.100
88	134.121.154.120
76	134.82.142.60

Total number of Rows: 10

"DoS ftpd globbing, is an attempt to crash the server by issuing a command like "LIST */../*../../*../../*". This will often overload the FTP server software, causing it to crash." Taken from Joe Ellis practical http://www.giac.org/practical/Joe_Ellis_GCIA.doc.

This alert requires that a tcp connect setup with the server and then above command issue.

Recommendations: Check the destination servers for vulnerable versions of the FTP and patch them. If these are public FTP servers they should placed on a public screen subnet so that the access is restricted.

ICMP Fragment Reassembly Time Exceeded

Statistics for Event: "ICMP Fragment Reassembly Time Exceeded"

Top 10 Destination Addresses

count	destip
1420	202.103.30.118
505	211.169.242.108
327	63.250.219.189
232	63.250.219.190
198	63.250.219.149
195	61.78.35.44
190	152.101.96.7
189	66.28.225.156
144	211.233.27.142
129	63.250.219.154

Total number of Rows: 10

Top 10 Source Addresses

count	souceip
1420	MY.NET.88.140
528	MY.NET.153.197
292	MY.NET.153.171
243	MY.NET.153.205
227	MY.NET.152.251
212	MY.NET.151.95
191	MY.NET.153.46
187	MY.NET.153.45
183	MY.NET.152.183
146	MY.NET.152.15

Total number of Rows: 10

Below is an extract from the logs:

```
03/31-16:58:29.104757 [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.152.171 -> 63.250.219.154
03/31-16:58:30.110475 [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.152.171 -> 63.250.219.154
03/31-16:58:32.114387 [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.152.171 -> 63.250.219.154
```

Here we will only look at the top 10 destinations as this ICMP error message is returned from a host that did not receive all the fragments advertised. Fragmentation is normally not very common on a network to start with. Attacks of this nature usually target IP stacks that can't handle abnormal fragmentation, with out a log of the traffic seen we can only take an educated guess. Usually when we see high amounts of fragmentation it usually means that some of the host are under some sort of fragmentation attack and we also that the errors are going to hosts outside the clients network. The nature of this type of attack has a high probability of the source being spoofed depending on the desired result being either to elude a Firewall or IDS or to crash a machine that can't handle out of spec fragmentation. Another point is that none of the source address here correlate with the list of out of spec source address in the "MISC Large UDP packet" Section.

Recommendations: If there is no firewall that can't handle fragmentation reassembly on the network perimeter then implement one, secondly investigate the source hosts if their state of trustworthiness is unknown or if there is a network configuration error. I have seen fragmentation in some VPN implementations where they have been misconfigured and were unable to take part in MTU path discovery.

Incomplete Packet Fragments Discarded

Statistics for Event: "Incomplete Packet Fragments Discarded"

Top 10 Destination Addresses

count	destip
3151	MY.NET.88.140
9	MY.NET.153.137
2	MY.NET.153.191
1	MY.NET.150.133

Total number of Rows: 4

Top 10 Source Addresses

count	souceip
3151	202.103.30.118
9	64.12.34.219
1	217.227.178.15
1	192.168.0.50
1	80.235.56.26

Total number of Rows: 5

This alert is caused when the sensor has not seen all the fragments.

We see 3151 of these type of alerts between the source 202.103.30.118 and destination MY.NET.88.140

04/08-15:23:51.038236 [**] Incomplete Packet Fragments Discarded [**] 202.103.30.118:0 -> MY.NET.88.140:0

04/08-15:24:04.685812 [**] Incomplete Packet Fragments Discarded [**] 202.103.30.118:0 -> MY.NET.88.140:0

Around this time there were problems associated with earlier versions of Snort's frag preprocessor and this preprocessor was superseded with the frag2 preprocessor to correct this problem. <http://www.ultraviolet.org/mail-archives/snort-users.2001/3408.html>. If we also look at the list of the source address that represent the "Out of Spec" packets from "Misc Large UDP packets" section, we see the none of the source addresses correlate.

Some software like "Netop" can also cause problems as it allows you to set the packet size to be far greater than the maximum MTU size, causing a lot of fragmentation to occur.

If software like Netop is being used why is coming from outside the internal network ?

Recommendations: Check the "snort.conf" file is not using the old frag preprocessor if it is, then use frag2 or upgrade to a newer version of snort. If the new version is installed you may need to investigate if there is a legitimate relationship between the two hosts. If no relationship exists then this host is probably being subjected to some sort of fragmentation attack. If it is an attack it is more likely to be a DOS that is tying up the system resources.

WEB-IIS view source via translate header

The following was taken from <http://www.whitehats.com/IDS/305>:

"Microsoft IIS 5.0 has a dedicated scripting engine for advanced file types such as ASP, ASA, HTR, etc. files. The scripting engines handle requests for these file types, processes them accordingly, and then executes them on the server.

It is possible to force the server to send back the source of known scriptable files to the client if the HTTP GET request contains a specialized header with 'Translate: f' at the end of it, and if a trailing slash '/' is appended to the end of the URL. The scripting engine will be able to locate the requested file, however, it will not recognize it as a file that needs to be processed and will proceed to send the file source to the client"

The event was possibly triggered by a rule such as:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS view source via translate header"; flags: A+; content: "Translate|3a| F"; nocase; reference:arachnids,305; reference:bugtraq,1578; classtype:web-application-activity; sid:1042; rev:3;)
```

False positives can be generated by legitimate requests from WebDAV.

All these events are coming from external IP Addresses going to only 3 destinations MY.NET.5.96 being the most popular, The bulk are multiple connections from the same sources.

```
$ cat alert.020??? | grep "WEB-IIS view source via translate header" | cut -d "]" -f 3- | sed -e 's:/ /g' | awk '{print $1," ",$4}' | sort -u | sort -k 2
```

```
68.55.200.56 MY.NET.150.220
```

```
24.81.71.153 MY.NET.150.83
```

```
12.78.130.17 MY.NET.5.96
```

```
.....  
68.55.47.100 MY.NET.5.96
```


68.55.56.152 MY.NET.5.96

Correlations:

Paul Farley has also seen this activity in his practical http://www.giac.org/practicals/Paul_Farley_GCIA.doc

Recommendations: If these are public web servers make sure that they are running the most up to date security patches and are located on a screened subnet, Also make sure that any scripts that are running on the web server are also secure.

ICMP Router Selection

Statistics for Event: "ICMP Router Selection"

Top 10 Destination Addresses

count	destip
1748	224.0.0.2
Total number of Rows: 1	

Top 10 Source Addresses

count	souceip
113	MY.NET.153.71
75	MY.NET.150.165
74	MY.NET.153.46
57	MY.NET.153.45
48	MY.NET.88.151
41	MY.NET.88.149
37	MY.NET.150.232
33	MY.NET.150.210
27	MY.NET.151.98
25	MY.NET.150.241

This event is described in RFC1256 <http://www.ietf.org/rfc/rfc1256.txt> and is considered to be ICMP Router discovery. Router Discovery enables hosts that are attached to a broadcast network to discover the IP addresses of neighboring routers. If a host is RFC1256 compliant it can be configured with a default gateway at the system boot process. If the default gateway is down then the host sends out an ICMP router discovery request using ICMP (type 10, Code 0) to the multicast address of 224.0.0.2. Routers that actually support RFC 1256 on the local network respond with a Router advertisement which enables the

host to choose the router as its new default gateway. Another source of information on how routers and host that are RFC1256 compliant can be found in TCP/IP illustrated Vol. 1, Chapter 9.6.

Recommendations: Looking at the alerts these events not only come from the local network where the sensors located but remote networks also. I would that the multicast addresses are not routed on the clients network as this type of traffic is only meant for local subnets.

WEB-MISC Attempt to execute cmd

This event is usually indicative that someone is trying to use the IIS Unicode vulnerability to execute cmd.exe on Microsoft platforms. It could also be someone looking for systems that have been compromised by one of the numerous worms that are out in the wild. Either way it is impossible to tell what exactly happened without examining the payload contents.

The source addresses are all external IP addresses going to internal web servers with multiple events for each source. This type of activity is consistent with worm or script kiddie activity.

03/31-04:02:36.693037 [**] WEB-MISC Attempt to execute cmd [**] 212.87.23.220:3569 -> MY.NET.151.114:80

03/31-04:03:18.995009 [**] WEB-MISC Attempt to execute cmd [**] 212.87.23.220:4800 -> MY.NET.153.219:80

03/31-04:03:19.007595 [**] WEB-MISC Attempt to execute cmd [**] 212.87.23.220:4801 -> MY.NET.153.220:80

Correlations:

http://www.giac.org/practices/Roland_Lee_GCIA.doc

Recommendations: Public web servers should be located in screened subnets, so that if compromised they amount of damage that can be caused is limited to only that subnet. All internal web servers should not be accessible to the public and ALL web servers should be running the latest security patches. VPNs also serve as great entry points into the internal network to release into the internal networks to release such worms and should be secured accordingly.

NMAP TCP ping!

When host are behind firewalls or other packet filtering devices you may not be able to receive and ICMP echo reply in response to your ICMP echo request. With NMAP you do a TCP Ping. From the command line it will look like "nmap -sP -PT<PORT> <HOST>". Nmap sends a TCP ACK to the host, if the host is up it will send a TCP "RESET" as it has no knowledge of any TCP connection or does not have a service listening on the specified port. See example below

16:41:03.563272 172.16.89.73.51357 > 172.16.89.140.25: . ack 1138444992 win 2048

16:41:03.563769 172.16.89.140.25 > 172.16.89.73.51357: R 1138444992:1138444992(0) win 0

The Signature for this rule is probably look like this:

alert tcp \$EXTERNAL_NET any -> \$HOME_NET any (msg:"SCAN nmap TCP"; flags:A; ack:0; reference:arachnids,28; classtype:attempted-recon; sid:628; rev:1;)

Here the Ack flag is set and has an Acknowledgement number of 0.

The traffic below looks like to be disguised as or is Kazaa/Web traffic.

03/31-19:27:08.298825 [**] NMAP TCP ping! [**] 65.193.73.247:80 -> MY.NET.153.191:1214

```
04/01-11:31:14.382740 [**] NMAP TCP ping! [**] 163.23.190.2:80 -> MY.NET.153.191:1214
04/01-11:50:25.155202 [**] NMAP TCP ping! [**] 160.124.224.254:80 -> MY.NET.153.191:1214
04/01-17:56:29.988284 [**] NMAP TCP ping! [**] 64.119.138.2:80 -> MY.NET.150.113:1214
```

If we look at the scan logs for just the first event we can see that MY.NET.153.191 has also been actively scanning other hosts it is also in the overall TOP 10 source addresses.

```
Mar 31 14:36:38 MY.NET.153.191:2547 -> 67.80.239.2:1214 SYN *****S*
Mar 31 14:36:32 MY.NET.153.191:2549 -> 207.94.106.3:1214 SYN *****S*
Mar 31 14:36:34 MY.NET.153.191:2550 -> 194.192.131.148:1214 SYN *****S*
```

Without a payload capture it is hard to draw any more conclusions than this.

Another question is, Why would you allow Kaaza traffic into internal network ?

Recommendations: A Truly Stateful Firewall would not allow this traffic in, and it is recommended that one is implemented to stop this kind of activity. Beware some firewalls used to and still do allow this type of access provided there is a rule that permits it. It is supposed to speed up connections, this problem has been corrected in Checkpoint Firewalls about 6 months ago. Also check if MY.NET.153.191 is authorized to conduct network auditing, otherwise it needs to be investigated further.

Watchlist 000222 NET-NCFC

Watch list are design to flag a hosts that have been involved in previous suspicious activity unfortunately snort exits as soon as a match is found. So no more information is available other than this host was seen again accessing some other network resources

Destination ports for this event are:

```
$ cat alert.020??? | grep "Watchlist 000222 NET-NCFC" | cut -d "]" -f 3- | sed -e 's:/ /g' | awk '{print $5}' | sort -u
1098, 1608, 1713, 1752, 1753, 1754, 1922, 2163, 2353, 2421, 2611, 2859, 3094, 3328, 3550, 3783, 4000, 4254, 4662, 80
```

There is not much interesting here, some file sharing applications and web server activity. The other high ports could be attributed to passive FTP or connections made in the opposite direction. Unless there is a packet capture so that it can be read back into snort it does not give much more information.

```
$ WATCHLIST=`cat alert.020??? | grep "Watchlist 000222 NET-NCFC" | cut -d "]" -f 3- | sed -e 's:/ /g' | awk '{print $1}' | sort -u`
```

```
$ echo $WATCHLIST
```

```
159.226.117.175 159.226.150.17 159.226.47.197 159.226.83.23 159.226.87.6
```

```
$ for IP in $WATCHLIST ; do cat alert.020??? | grep -v Watchlist | grep $IP >> watch.txt ; done
```

```
$ cat watch.txt
```

No Results !

Looking at the scan logs it only appears that 159.226.83.23 is using edonkey which is another file sharing application to from an internal host and has probably been turned off by its owner.

```
Apr 4 19:35:35 MY.NET.150.143:1098 -> 159.226.83.23:4662 SYN *****S*
Apr 4 19:35:36 MY.NET.150.143:1098 -> 159.226.83.23:4662 SYN *****S*
Apr 4 20:15:37 MY.NET.150.143:1608 -> 159.226.83.23:4662 SYN *****S*
```

Apr 8 00:36:46 MY.NET.150.143:4254 -> 159.226.83.23:4662 SYN *****S*

Apr 5 16:04:44 MY.NET.153.164:2353 -> 159.226.87.6:6346 SYN *****S*

Recommendations: If this host from the 159.x.x.x net block are a concern, then effective perimeter defense should be installed limiting what applications these source hosts have access to or cut them off all together

WEB-IIS _vti_inf access & WEB-FRONTPAGE _vti_rpc access

The signature that trigger these events probably looked like this:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS _vti_inf access"; flags: A+; uricontent: "_vti_inf.html"; nocase; classtype: web-application-activity; sid:990; rev:3;)
```

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-FRONTPAGE _vti_rpc access"; flags: A+; uricontent: "/_vti_rpc"; nocase; reference: bugtraq,2144; classtype: web-application-activity; sid:937; rev:4;)
```

As the signatures indicate this is web application activity and I would not consider a threat. Legitimate web activity can cause this type event to be generated. It is time to be concerned when you see these events in groups of 3 or more from the same source, when see this pattern it is usually indicative of someone probing your web server in order to gain more information. Below we see two events directly after each other.

03/31-01:10:08.211911 [**] SMB Name Wildcard [**] MY.NET.11.6:137 -> MY.NET.152.21:137

03/31-01:10:16.550110 [**] WEB-IIS _vti_inf access [**] 172.155.198.189:1826 -> MY.NET.5.96:80

03/31-01:10:18.053253 [**] WEB-FRONTPAGE _vti_rpc access [**] 172.155.198.189:1827 -> MY.NET.5.96:80

03/31-01:10:27.033907 [**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.152.45-> MY.NET.11.7

Only with examining the contents of the payload can make sure of what actually happened.

Correlations: Other people have seen this activity

<http://archives.neohapsis.com/archives/incidents/2001-07/0098.html>

http://www.giac.org/practical/Matthew_Fiddler_GCIA.doc

http://www.giac.org/practical/Joe_Ellis_GCIA.doc

Recommendations: Public web servers should be located on screened subnet so that if compromised that the damage is contained to that subnet without providing a steppingstone to the rest of the network. Install the most up to date security patches. If these servers are not IIS servers than tune the signatures to ignore these servers for this alert.

Null scan!

A Null scan is a crafted TCP packet is with none of the flags set in the TCP header. It is used for reconnaissance to map networks, since no flags are set a simple firewall or router may let this packet through because they might only check for only valid combinations of flags to be set according to the RFCs.

This activity does not occur in the naturally and as stated before it is a crafted packet. This should certainly not be seen on the internal network.

Below is a signature that may have caused this event:

alert tcp \$EXTERNAL_NET any -> \$HOME_NET any (msg:"Null scan!";flags:0; seq:0; ack:0; reference:arachnids,4;)

More information can be found at <http://www.whitehats.com/IDS/4>.

Below is an extract from the logs:

```
04/01-13:06:51.596993  [**] Null scan! [**] MY.NET.186.16:23 -> MY.NET.150.137:1987
04/01-13:13:38.452484  [**] Null scan! [**] MY.NET.186.16:23 -> MY.NET.150.137:1987
04/01-13:23:38.731782  [**] Null scan! [**] MY.NET.186.16:23 -> MY.NET.150.137:1987
04/01-13:28:38.862431  [**] Null scan! [**] MY.NET.186.16:23 -> MY.NET.150.137:1987
```

What is alarming is the that the following three internal hosts are responsible for a good portion of the alerts

```
$ cat alert.020??? | grep "Null scan" | cut -d "]" -f 3- | cut -d ":" -f 1 | sort -u | grep MY
```

MY.NET.186.16

MY.NET.226.90

MY.NET.253.10

Correlations: Other Sans Students have seen this activity Lorraine Weaver http://www.giac.org/practical/Lorraine_Weaver_GCIA.zip

Recommendations: The three internal hosts, unless they authorized network scanning stations should be investigated to see if they have been compromised. Effective perimeter defense should be implemented so that this type of traffic does not make into the internal network.

ICMP Echo Request Windows

Top 10 Source Addresses

count	souceip
149	MY.NET.5.87
39	MY.NET.88.226
16	MY.NET.88.210
9	MY.NET.88.217
7	MY.NET.88.134
6	MY.NET.88.197
6	MY.NET.88.192
5	MY.NET.88.191
5	MY.NET.88.251
5	MY.NET.150.133

Total number of Rows: 10

This event describes ICMP echo request coming form windows host. Looking at the number of alerts and also the top sources for this alert, I would consider this activity fairly benign.

Recommendations: This would be a false positive, although this signature could be used to discover network mapping efforts using ICMP or covert channels like Tribal Flood network, it is prone to generating lots of false positives. Either take it out or live with the noise it generates.

Possible trojan server activity

This indicates the possibility of a Trojan backdoor that may exist on the network. While port 27374 is indicative that Sub 7 may exist, The signature could generate false positives if not written correctly.

03/31-18:55:16.780960 [**] Possible trojan server activity [**] MY.NET.70.177:27374 -> MY.NET.5.83:8903

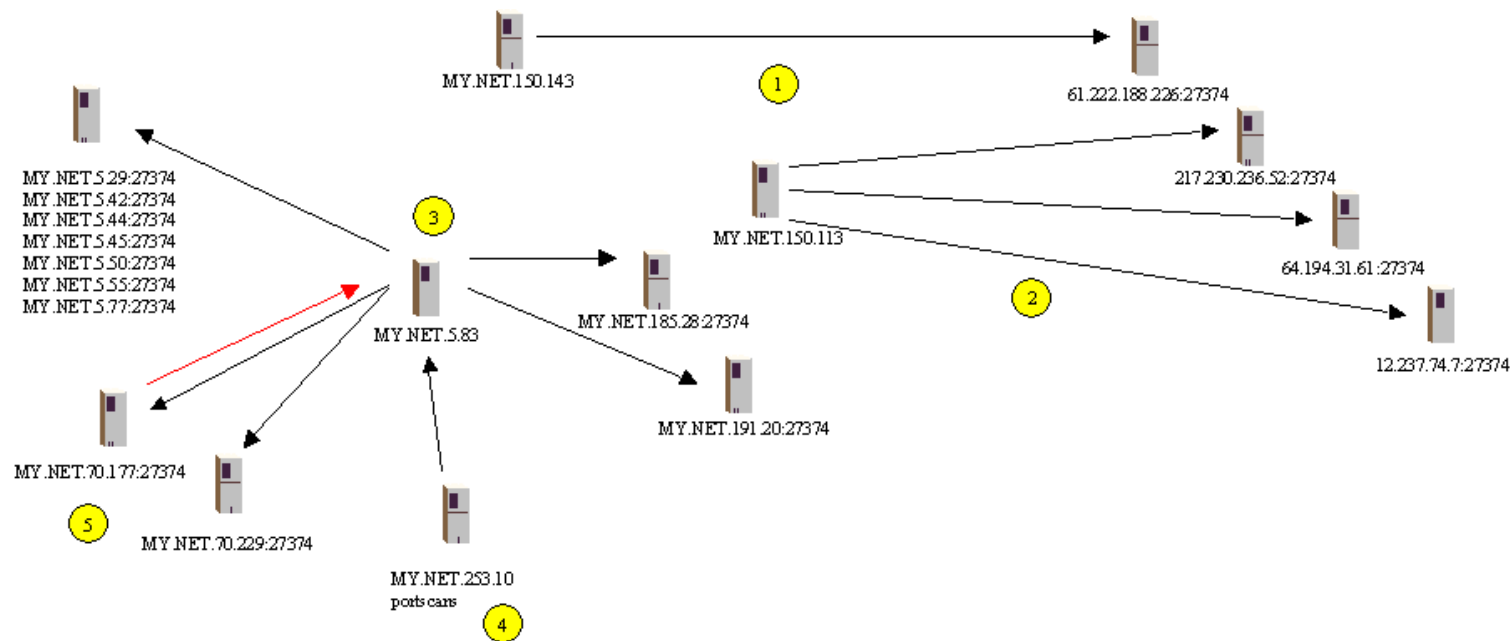
03/31-18:55:16.781321 [**] Possible trojan server activity [**] MY.NET.5.83:8903 -> MY.NET.70.177:27374

03/31-18:55:16.782640 [**] Possible trojan server activity [**] MY.NET.70.177:27374 -> MY.NET.5.83:8903

Here you can see the alert generated with the destination port of 8903 in line 1 and 3, this could be caused from what the internal network is defined as in the "snort.conf" file.

To get a better picture it may be useful to draw a "linksys" graph as to ascertain what is going on.

Below is such a graph for focusing on the Trojan event, It is drawn with arrows indicating the traffic flow toward the host that may have sub 7 installed. I have also created yellow markers and will attempt to describe the activity happening at these points.



Points 1 and 2 show possible sub 7 activity between internal and external hosts. What you actually see here is that the activity is more likely to be caused by not so malicious network activity from applications such as eDonkey and Kazaa. This is indicated by the presence of port 1214 and 4662 as the other port in the port pair for the connections that may be involved in sub 7 activities.

```
$ cat trojan.txt | egrep -e "(^[0-9]| > [0-9])"
217.230.236.52:27374 -> MY.NET.150.113:1214
.....<snip>.....
64.194.31.61:27374 -> MY.NET.150.113:1214
.....<snip>.....
61.222.188.226:27374 -> MY.NET.150.143:4662
61.222.188.226:27374 -> MY.NET.150.143:4662
```

Point 3 shows MY.NET.5.83 being the source of lots of connections to other host on port 27374. The extract below shows that this host is something to be concerned about as these connections can not be attributed to other network activity. The next question is, if this is Trojan activity I would expect other connections to this host from an external host or a connection from an external host to another internal machine and then on to this host.

```
MY.NET.70.177:27374 -> MY.NET.5.83:8903
MY.NET.5.83:8903 -> MY.NET.70.177:27374
....<snip>.....
MY.NET.5.45:27374 -> MY.NET.5.83:7938
MY.NET.5.83:7938 -> MY.NET.5.45:27374
```

Points 4 and 5 are trying to test the theory that other connections coming from external sources to MY.NET.5.83. However this turns up negative for the theory of Sub 7 connection being made from an external IP address or from another internal host that had an external connection. MY.NET.253.10 seems to be conducting a lot of port scans overall, these were detected with the Snort pre-processor. It is also conducting a lot of NMAP/HPING2 type scans towards MY.NET.5.83. The Connection indicated with a red arrow from MY.NET.70.177 to MY.NET.5.83 is a connection on port 161, which is used for SNMP. The interesting thing that it does not look like a typical SNMP connection as the source port is here it is not 161 but 1072. See below extract.

```
03/31-07:37:58.345275  [**] SNMP public access [**] MY.NET.70.177:1072 -> MY.NET.5.143:161
03/31-07:37:58.345806  [**] SNMP public access [**] MY.NET.70.177:1072 -> MY.NET.5.141:161
03/31-07:47:25.442755  [**] spp_portscan: portscan status from MY.NET.70.177: 11 connections across 5 hosts: TCP(8), UDP(3) [**]
```

Recommendations: Investigate MY.NET.5.83 for presence of the SUB-7 trojan or Ramen worm. Also check MY.NET.253.10 and MY.NET.70.177 to establish if they are Network auditing stations. If they are not then they need to be further investigated to find the reasons for this abnormal traffic. Furthermore effective perimeter defense will stop connections from the outside for this Trojan activity. The signatures could also be tuned so that the attacks are more accurately detected with a clear definition of what is the internal network and what the external network is.

WEB-CGI scriptalias access

If there is a poorly configured Apache web server with the script alias directory that is configured directly under the document root. It may allow an attacker to download the CGI scripts on this web server. This allows the attacker to further analyze them for possible weaknesses and compromise the web server at a later time if one is found. More information can be found at <http://www.whitehats.com/IDS/227>.

The signature that caused the event probably looked like this:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI scriptalias access"; flags:A+; uricontent: "///"; reference:cve,cve-1999-0236; reference:bugtraq,2300; reference:arachnids,227; classtype:attempted-recon; sid:873; rev:3;)
```

Below is an extract from the logs:

```
04/01-11:14:35.311044  [**] WEB-CGI scriptalias access [**] 68.55.176.169:64699-> MY.NET.5.96:80
04/01-11:14:36.248991  [**] WEB-CGI scriptalias access [**] 68.55.176.169:64509-> MY.NET.5.96:80
04/01-11:15:08.662733  [**] WEB-CGI scriptalias access [**] 68.55.176.169:65133-> MY.NET.5.96:80
```

165 of the 225 total events are from 68.55.176.169 and all the events are directed towards the MY.NET.5.96.

68.55.176.169 belongs to a cable network in the US. It also has not been involved in any other incidents

Comcast Cable Communications, Inc. JUMPSTART-1 (NET-68-32-0-0-1)

68.32.0.0 - 68.63.255.255
Comcast Cable Communications, Inc. JUMPSTART-BALTIMORE-A (NET-68-54-80-0-1)
68.54.80.0 - 68.55.255.255

Correlations: Other studying their SANS GCIA have also seen this type of event

http://www.giac.org/practical/safka_gcia.doc

http://www.giac.org/practical/Matthew_Fiddler_GCIA.doc

Recommendations: Check MY.NET.5.96 that it is configured properly and that all the latest security patches are applied. If it is a public web server it should be placed on a screened subnet, so that if it is compromised the chance it can be used as a launching point into the rest of the internal network is limited. Also put 68.55.176 on a watchlist.

SCAN Proxy attempt & INFO - Possible Squid Scan

Someone is looking for Open Proxy that might be located on the internal network. If they find one that is configured for anonymous access they can use it to mask their activities while launching various attacks through that proxy. Here we see all source addresses from external networks. There should be no reason that external networks should have this type of access to the clients internal network.

The signatures may have looked like the ones below:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 3128 (msg:"SCAN Squid Proxy attempt"; flags:S; classtype:attempted-recon; sid:618; rev:2;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 1080 (msg:"SCAN SOCKS Proxy attempt"; flags:S; reference:url,help.undernet.org/proxyscan/;
classtype:attempted-recon; sid:615; rev:3;)
```

Below is an extract from the logs:

```
04/01-17:06:44.030086 [**] SCAN Proxy attempt [**] 146.20.33.71:2074 -> MY.NET.152.46:1080
04/01-17:06:44.864199 [**] SCAN Proxy attempt [**] 146.20.33.71:2106 -> MY.NET.152.46:8080
03/31-16:25:09.096059 [**] INFO - Possible Squid Scan [**] 217.115.140.87:4703-> MY.NET.152.171:3128
03/31-16:27:23.370838 [**] INFO - Possible Squid Scan [**] 216.152.64.163:33561 -> MY.NET.152.171:3128
```

All events come from external IP addresses

Recommendations: Implement effective perimeter security so that this traffic from external network never makes into the internal network.

INFO FTP anonymous FTP

Below is a Extract from the log:

```
03/31-14:12:31.485691 [**] INFO FTP anonymous FTP [**] 194.38.83.245:3632 -> MY.NET.150.243:21
03/31-14:12:33.907608 [**] INFO FTP anonymous FTP [**] 194.38.83.245:3632 -> MY.NET.150.243:21
03/31-14:12:37.097582 [**] INFO FTP anonymous FTP [**] 194.38.83.245:3632 -> MY.NET.150.243:21
```

Top 10 Destination Addresses

count	destip
-------	--------

```
-----
17          MY.NET.150.59
16          MY.NET.150.147
15          MY.NET.150.83
15          MY.NET.150.197
15          MY.NET.153.219
14          MY.NET.150.243
11          MY.NET.150.101
11          MY.NET.150.195
10          MY.NET.150.231
10          MY.NET.153.220
```

Total number of Rows: 10

As this alert suggests "INFO FTP anonymous FTP" is actually anonymous FTP server access. All the events come from external IP addresses. While I would normally not consider this hostile activity normally, the question is, are these legitimate Anonymous FTP servers ?

Recommendations: Check if these are legitimate FTP servers, if they are then this is probably OK. However since all the connections are coming from external networks I would move these servers to a screened DMZ with adequate perimeter defense so that they don't traverse the internal network if this is not already the case. Also make sure that these servers are also running a chroot'd most current version of the FTP daemon.

ICMP Destination Unreachable (Communication Administratively Prohibited)

This event is generated when a host tries to send traffic to a host behind a router or a firewall that is blocked. Routers and some firewalls will respond with an ICMP Destination Unreachable (Communication Administratively Prohibited). This information can be used by an attacker to map ACL's (Access Control List) installed on a Router or Firewall. If the payload is available you can see what traffic caused the event as this message is only sent by the Router or Firewall itself.

The signature that generated this event probably looked like this:

```
alert icmp any any -> any any (msg:"ICMP Destination Unreachable (CommunicationAdministratively Prohibited)"; itype: 3; icode: 13; classtype:misc-activity; sid:485; rev:2;)
```

Below is an extract from the logs:

```
03/31-06:44:21.627697  [**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**] MY.NET.150.1 -> MY.NET.150.24
03/31-06:44:27.635005  [**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**] MY.NET.150.1 -> MY.NET.150.24
03/31-10:23:50.931437  [**] ICMP Destination Unreachable (Communication Administratively Prohibited) [**] MY.NET.150.1 -> MY.NET.150.24
```

All traffic is between MY.NET.150.1 and MY.NET.150.24, MY.NET.150.1 being the source address. This is consistent with the theory that this message is being generated by a router or firewall. since the addresses for these devices is either at the beginning or end of the range of the subnet. If we had a capture

of the payload we could see what caused this error to be generated. I would guess in this case it may actually be a network configuration error on MY.NET.150.24 since it is on the same subnet as the Router/Firewall.

Recommendations: If the payload is available, check it for the cause of the error. Also check the host MY.NET.150.24 for network configuration errors.

IDS552/web-iis_IIS ISAPI Overflow ida nosize

This event indicates that someone has tried to exploit a vulnerability in Microsoft IIS web server. An unchecked buffer in the IIS Index Server ISAPI extension could enable an attacker to gain system access remotely to the Web Server. More Information can be found at <http://www.whitehats.com/IDS/552>.

Below are two examples of possible signatures:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS ISAPI .ida attempt"; uricontent:".ida?"; nocase; dsize:>239; flags:A+;
reference:arachnids,552; reference:bugtraq,1065; reference:cve,can-2000-0071; classtype:web-application-attack; sid:1243; rev:4;)
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS ISAPI .ida access"; uricontent:".ida"; nocase; flags:A+;
reference:arachnids,552; reference:cve,can-2000-0071; reference:bugtraq,1065; classtype:web-application-activity; sid:1242; rev:4;)
```

Below is an extract from the logs:

```
04/07-12:05:19.795136 04/07-12:12:52.354158 04/07-12:29:11.488950
[**] IDS552/web-iis_IIS ISAPI Overflow ida nosize [**] 216.175.67.90:1294 -> MY.NET.150.195:80
[**] IDS552/web-iis_IIS ISAPI Overflow ida nosize [**] 61.120.56.149:3260 -> MY.NET.150.220:80
[**] IDS552/web-iis_IIS ISAPI Overflow ida nosize [**] 216.170.21.119:23202 -> MY.NET.5.243:80
```

Correlations: The following SANS GIAC students have seen this activity

http://www.giac.org/practical/Rick_Yuen_GCIA.doc
http://www.giac.org/practical/Joe_Ellis_GCIA.doc

Recommendations: Make sure that any IIS servers that are running are patched with the latest security updates. If they are public servers they should be located on screened DMZ so that if compromised that amount of damage that can be done is limited. If they are not IIS servers adjust the signatures to take them into account.

ICMP traceroute

Top 10 Destination Addresses

count	destip

76	MY.NET.152.1
15	MY.NET.88.129
3	MY.NET.150.243
3	MY.NET.98.180
2	MY.NET.1.3
2	MY.NET.151.1

Total number of Rows: 6

Top 10 Source Addresses

count	souceip
29	MY.NET.152.20
8	MY.NET.152.180
6	MY.NET.152.250
4	MY.NET.152.247
4	MY.NET.152.252
4	MY.NET.152.249
3	MY.NET.88.140
3	MY.NET.169.246
3	MY.NET.153.111
2	MY.NET.152.139

Total number of Rows: 10

“ICMP Traceroute” may indicate some network mapping is taking place or some one is trying to debug a connection to a certain host. All this traffic is on the internal network and would consider the later explanation more likely. ICMP traceroute is more likely to be a Microsoft platform using the Tracert program as traceroute is Unix program and uses high UDP port numbers to perform the same function. However MTR(Matt’s traceroute) is also a unix program that uses ICMP to perform a traceroute function. The advantage of using ICMP is that it some firewalls allow ICMP through because it has many more uses than traceroute in that it can report network error conditions to a source host.

Recommendations: I would consider this activity benign. But it maybe in idea to put the MY.NET.152.20 onto a watch list and see if there is any further activity.

WEB-CGI rsh access & WEB-CGI ksh access

If perl, sh, csh or other interpreters are installed in the cgi-bin directory on a web server, it could allow attackers to execute arbitrary commands remotely

The signatures for these events may have looked like this:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI rsh access"; flags:A+; uricontent:"/rsh"; nocase; reference:cve,can-1999-0509; reference:url,www.cert.org/advisories/ca-1996-11.html; classtype:attempted-recon; sid:868; rev:4;)
```

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI ksh access"; flags:A+; uricontent:"/ksh";nocase; reference:url,www.cert.org/advisories/ca-1996-11.html; reference:cve,can-1999-0509; classtype:attempted-recon; sid:865; rev:3;)
```

Below is an extract from the logs:

```
04/08-00:09:56.487608 [**] WEB-CGI rsh access [**] 209.244.230.191:3685 -> MY.NET.5.96:80
```

```
04/08-00:10:10.471416 [**] WEB-CGI rsh access [**] 209.244.230.191:3685 -> MY.NET.5.96:80
04/08-00:10:14.439645 [**] WEB-CGI rsh access [**] 209.244.230.191:3686 -> MY.NET.5.96:80
04/01-12:46:00.330968 [**] WEB-CGI ksh access [**] 207.172.11.147:41485 -> MY.NET.5.96:80
04/01-12:46:08.405409 [**] WEB-CGI ksh access [**] 207.172.11.147:42384 -> MY.NET.5.96:80
04/01-12:46:11.155457 [**] WEB-CGI ksh access [**] 207.172.11.147:42796 -> MY.NET.5.96:80
```

All 88 events for "WEB-CGI rsh access" are from source 209.244.230.191 and all 74 events for "WEB-CGI ksh access" are from source 207.172.11.147. 207.172.11.147 has been observed in other types of web server reconnaissance with following activities listed below.

```
IDS475/web-iis_web-webdav-propfind
WEB-CGI ksh access
WEB-FRONTPAGE _vti_rpc access
WEB-IIS _vti_inf access
WEB-IIS view source via translate header
```

All events for both alerts have targeted MY.NET.5.96

Recommendations: MY.NET.5.96 should have the latest security patches installed. It should also be checked to see that there are no shell interpreters install in the cgi-bin directory. If it is a public web server it should be located on a screen subnet so that if compromised the damage can be limited to that host or screened subnet. 207.172.11.147 should be placed on a watch list for further observation.

WEB-MISC 403 Forbidden

This is an error "403 Forbidden" is generated by a web server. This is an error that occurs when a web server can't access the file you requested due to a file permission. It also happens when if your account is suspended. See <http://faq.site5.com/read.php?article32>.

Below is an extract from the logs:

```
03/31-05:13:28.772463 [**] WEB-MISC 403 Forbidden [**] MY.NET.150.59:80 -> 211.93.8.74:22705
03/31-05:13:36.156876 [**] WEB-MISC 403 Forbidden [**] MY.NET.150.59:80 -> 211.93.8.74:22894
03/31-10:50:24.086196 [**] WEB-MISC 403 Forbidden [**] MY.NET.5.96:80 -> 64.12.97.11:19954
```

Each client, being the hosts seen as the destination in these logs are all external. Overall there is no more than a maximum of 10 attempts from any one of the clients to and they seem to be directed at no more than two of the following web servers MY.NET.150.59, MY.NET.5.92, MY.NET.5.96.

This type of alert can also indicate that someone may be performing another web application attack for example a Unicode attack.

If we correlate the hosts that are causing the internal web serves to generate this event we might come up with some information about hosts seen in the Unicode attack.

```
$ cat alert.020??? | grep "WEB-MISC 403 Forbidden" > forbidden.txt
$ ADDRESSES=`cat forbidden.txt | cut -d ">" -f 2 | sed -e "s/^ //" |cut -d ":"-f 1 | sort -u`
$ echo $ADDRESSES
12.91.163.139 12.91.163.151 130.227.199.190 131.118.250.187 131.118.250.188 131.
```

```

50.151.8 172.131.124.8 172.132.236.128 198.26.130.37 204.210.31.231 211.100.25.1
98 211.93.8.74 216.35.116.92 216.35.116.93 216.45.81.150 24.28.217.25 63.125.55.
223 64.12.96.200 64.12.97.11 65.100.92.136 66.200.114.148 68.3.150.2 68.50.28.15
3 68.55.19.80
$ for IP in $ADDRESSES ; do echo ; echo "$IP" ; echo ; cat alert.020??? | grep "IIS Unicode attack detected" | grep "]" $IP" | wc -l ; done >
403inUnicode2.txt

```

We end up with 130.227.199.190 causing 26 Unicode events and 211.93.8.74 causing 14 Unicode events.

Unique events for 130.227.199.190:

```

$ cat alert.020??? | cut -d "]" -f 2- | grep "130.227.199.190" | cut -d "[" -f 1 | sort -u
WEB-IIS Unauthorized IP Access Attempt
WEB-MISC 403 Forbidden
WEB-MISC Attempt to execute cmd
spp_http_decode: IIS Unicode attack detected

```

Unique events for 211.93.8.74:

```

$ cat alert.020??? | cut -d "]" -f 2- | grep "211.93.8.74" | cut -d "[" -f 1 | sort -u
WEB-MISC 403 Forbidden
WEB-MISC Attempt to execute cmd
spp_http_decode: IIS Unicode attack detected

```

Correlations: Edward Peck has also seen this activity. http://www.giac.org/practical/Edward_Peck_GCIA.doc

Recommendations: Apart from checking the web server logs as to what the users were actually trying access, the 2 source hosts 130.227.199.190 and 211.93.8.74 should be blocked by an ACL on router or firewall at the network perimeter and the network owners notified that these hosts are used in malicious behavior. The information on the networks and who owns them can be obtained by searching the “whois” databases for RIPE, ARIN and APNIC.

FTP CWD / - possible warez site

This event occurs if when a user opens a session to an FTP server and types the command “cd / “ to go to the root directory of a ftp server. However this is prone to a lot of false positives as there are a lot of NT FTP servers out there that have world write able “/” dirs.

Below is an extract from the logs:

```

03/31-14:12:43.943527  [**] FTP CWD / - possible warez site [**] 194.38.83.245:3632 -> MY.NET.150.243:21
03/31-14:12:44.091103  [**] FTP CWD / - possible warez site [**] 194.38.83.245:3632 -> MY.NET.150.243:21
03/31-14:12:48.066233  [**] FTP CWD / - possible warez site [**] 194.38.83.245:3634 -> MY.NET.151.114:21

```

All 54 attempts are from 194.38.83.245. The destinations are listed as follows:

```

MY.NET.150.101

```

MY.NET.150.147
MY.NET.150.195
MY.NET.150.197
MY.NET.150.231
MY.NET.150.243
MY.NET.150.59
MY.NET.150.83
MY.NET.150.84
MY.NET.151.114
MY.NET.153.219
MY.NET.153.220

Recommendations: If these are legitimate Public web servers than the configuration should be check so that the / dir is not world write able, this should be done even if they are internal. If they are not, there should be effective border protection in place so that they can't be accessed by the public.

WEB-MISC compaq nsight directory traversal

"This event indicates that an intruder has attempted to exploit a directory traversal vulnerability in the Compaq Web Management Agent. This allows a remote attacker to read arbitrary files". Taken from <http://www.whitehats.com/IDS/244>.

The signature may have looked like the one below:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 2301 (msg:"WEB-MISC compaq nsight directory traversal"; content: "../"; reference:arachnids,244;
reference:cve,CVE-1999-0771;)
```

Below is an extract from the logs:

```
03/31-16:02:32.219211 162.129.44.33:8765 -> MY.NET.152.165:2301
03/31-16:02:32.220505 162.129.44.33:8765 -> MY.NET.152.165:2301
03/31-16:02:32.221733 162.129.44.33:8765 -> MY.NET.152.165:2301
```

Correlations: Other Sans students have seen this event:

http://www.giac.org/practical/Matthew_Fiddler_GCIA.doc

http://www.giac.org/practical/David_Jenkins_GCIA.doc

All events are from external networks !

Recommendations: Access from external IP address to management internal management interfaces should never happen. Implement effective boundary protection so that this does not occur.

EXPLOIT x86 NOOP

Top 10 Destination Addresses

count	destip
-------	--------

10	MY.NET.150.106
4	MY.NET.88.140
3	MY.NET.153.164
3	MY.NET.152.171
2	MY.NET.153.174
1	MY.NET.153.211
1	MY.NET.150.165
1	MY.NET.153.152
1	MY.NET.150.143
1	MY.NET.153.194

Total number of Rows: 10

Top 10 Source Addresses

count	souceip
10	216.51.18.131
7	207.46.177.148
3	211.219.153.49
2	216.52.138.18
2	63.214.22.67
2	63.121.98.69
2	63.251.52.75
2	205.138.230.234
1	63.250.219.154
1	63.250.219.153

Total number of Rows: 10

The destination ports for "EXPLOIT x86 NOOP" are a little misleading as they are not well known ports.

04/01-14:27:43.037842 [**] EXPLOIT x86 NOOP [**] 63.214.22.67:80 -> MY.NET.88.140:3883

04/01-14:35:49.499657 [**] EXPLOIT x86 NOOP [**] 63.121.98.69:80 -> MY.NET.88.140:4004

04/01-14:35:49.500890 [**] EXPLOIT x86 NOOP [**] 63.121.98.69:80 -> MY.NET.88.140:4004

If we take a closer look we see that the other port in the connection pair is port 80.

Possible snort signatures for this alert could be:

alert ip \$EXTERNAL_NET any -> \$HOME_NET \$SHELLCODE_PORTS (msg:"SHELLCODE x86 NO


```
OP"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; depth: 128; reference:arachnids,181; classtype:shellcode-detect; sid:648; rev:5;)
or
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86 NO
OP"; content:"|61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61|"; classtype:shellcode-detect; sid:1394; rev:3;)
```

I have observed this signature in the past creating a lot of false positives due to URL's being written in Unicode for example a small 'a' could be written as 61. Without a capture of the payload it is hard to make a judgment on whether this is a false positive or not.

Recommendations: If no payload is available check that that port 80 is not included in the \$SHELLCODE_PORTS variable. Also check what has been defined in as \$HOME_NET so that false positives are kept to a minimum.

EXPLOIT NTPDX buffer overflow

Top 10 Source Addresses

count	souceip
8	64.232.138.142
5	63.250.205.34
5	160.79.2.67
3	64.124.157.16
2	64.124.157.10
2	166.90.148.196
2	160.79.2.66
2	66.28.14.37
1	63.250.205.3
1	63.250.205.44

Total number of Rows: 10

Around August 2001 an NTPD buffer overflow was publicized

http://www.securiteam.com/unixfocus/NTPD_vulnerable_to_a_remotely_exploitable_buffer_overflow_readvar.html. The buffer overflow occurs when building a response to a query with a large readvar argument. The shellcode executed must be less than 70 bytes otherwise the destination buffer is damaged. This makes the vulnerability difficult to exploit but not impossible.

All the addresses are from the external network. NTP is a UDP based protocol which makes the source address easily able to be spoofed.

Recommendations: Implement effective perimeter defense to stop this traffic from entering the internal network. Check that all the destination addresses are not vulnerable to this type of attack if they are check the systems for any signs of intrusion and/or patch them.

Queso fingerprint

The Queso finger print occurs when the 2 reserved bits and the SYN flag is set in a TCP packet. The idea is that different operating systems will respond differently with illegal options set in the TCP flags. The response from the remote operating system can often give away its identity.

The other reason why this type of activity might be seen is that Linux systems from around the beginning of 2002 started making use of these reserved bits. These reserved bits can be used for Error Congestion Notification (ECN)

Below is an extract from the logs:

```
04/03-04:46:36.965943  [**] Queso fingerprint [**] 193.2.132.70:59632 -> MY.NET.153.170:6346
04/03-08:42:59.777707  [**] Queso fingerprint [**] 217.80.78.17:51580 -> MY.NET.150.143:4662
04/03-09:13:23.785469  [**] Queso fingerprint [**] 217.80.78.17:52113 -> MY.NET.150.143:4662
```

This behavior can be correlated in the OOS logs to verify the reserved bits being set:

```
04/03-16:20:16.363489 217.80.78.17:58009 -> MY.NET.150.143:4662
TCP TTL:53 TOS:0x0 ID:49037 DF
21S***** Seq: 0x4C4086FB Ack: 0x0 Win: 0x16B0
TCP Options => MSS: 1412 SackOK TS: 90585462 0 EOL EOL EOL EOL
```

Now lets take a closer look at this activity:

```
$ cat alert.020??? | grep "Queso fingerprint" | cut -d "]" -f 3- > temp.txt
$ cat temp.txt | sed -e 's/-> /:/g' | cut -d ":" -f 1,3- | sort -u | sed -e 's/:-> /'
193.2.132.70 -> MY.NET.153.170:6346
202.153.244.62 -> MY.NET.150.83:80
213.152.32.42 -> MY.NET.153.175:6346
217.235.147.155 -> MY.NET.153.160:6346
217.80.78.17 -> MY.NET.150.143:4662
24.208.197.119 -> MY.NET.152.244:6346
24.208.197.119 -> MY.NET.153.171:6346
24.83.3.75 -> MY.NET.150.226:80
62.57.26.67 -> MY.NET.153.160:6346
68.66.64.71 -> MY.NET.153.182:6346
```

The destination ports that come from this event are 80(http), 6346(Gnutella) and 4662(eDonkey).

If we look at one of these IP addresses, we find out that it belongs to a broadband provider called Road Runner <http://www.rr.com/rdrun/>

```
[~]# whois -h whois.arin.net 24.208.197.119
OrgName:   Road Runner
OrgID:     RRMA
NetRange:  24.208.0.0 - 24.211.31.255
CIDR:      24.208.0.0/15, 24.210.0.0/16, 24.211.0.0/19
NetName:   RR-CENTRAL-3BLK
```

TechHandle: ZS30-ARIN
TechName: ServiceCo LLC
TechPhone: +1-703-345-3416
TechEmail: abuse@rr.com

This activity is more likely hosts using the reserved bits for Error Congestion Notification as users on able network will be using all available bandwidth to search and download files.

Other people have also seen this activity and can be referenced in Lorraine Weavers practical http://www.sans.org/practicles/Lorraine_Weaver_GCIA.zip
For more information on OS fingerprinting you can refer to <http://www.insecure.org/nmap/nmap-fingerprinting-article.html>.

Recommendations: Implement a secured screened subnet to provide internet services for internal hosts. Do not allow connections to be made directly to the internal network so as the limit the amount of information available to a potential attacker. I would probably put these source addresses on a watch list as well.

Attempted Sun RPC high port access

Below is an extract from the logs:

```
04/01-16:08:12.054886 [**] Attempted Sun RPC high port access [**] MY.NET.6.50:1029 -> MY.NET.153.173:32771
04/01-16:50:29.175152 [**] Attempted Sun RPC high port access [**] MY.NET.6.60:33792 -> MY.NET.153.202:32771
04/01-16:50:29.183945 [**] Attempted Sun RPC high port access [**] MY.NET.6.60:33792 -> MY.NET.153.202:32771
```

Investigating the logs further

```
$ cat alert.020??? | grep "Attempted Sun RPC high port access" | cut -d "]" -f 3- | cut -d ":" -f 1 | grep -v MY
```

No Results !

```
$ cat alert.020??? | grep "Attempted Sun RPC high port access" | cut -d "]" -f 3- | cut -d " " -f 4 | grep -v MY
```

No Results !

The Sun RPC service is an alternative port mapper service. This is used to find other services that may be running on a remote host. This is considered to be one of the Top 10 causes of security incidents. It should not be accessible or used by any host that is exposed to the Internet as many flaws exist in this protocol. A good article can be found at <http://www.sans.org/top20/#U1> describing the various weaknesses.

This alert can also produce a lot of false positives if sensor is not tuned.

I have seen this alert flagged regularly when in a Unix environment that are running backup services like Legato which use this service to find which port the back up service agent is running on a remote host. Here we see all the connections originating from an internal hosts and this could be a plausible explanation for this activity.

Recommendations: Make sure that these services are not available from the Internet. Make sure these services are only running on machines that require it and have the latest patches. Tune the sensors to into account this type of access if it is valid behavior.

MISC traceroute

I would say that this alert is caused by something else and is not a "MISC traceroute"

```
04/03-05:08:42.332897 [**] MISC traceroute [**] 192.204.106.2:24068 -> MY.NET.153.191:1214
04/03-05:09:16.822483 [**] MISC traceroute [**] 192.204.106.2:23815 -> MY.NET.153.191:1214
04/03-05:10:12.814824 [**] MISC traceroute [**] 192.204.106.2:24068 -> MY.NET.153.191:1214
```

All 35 events are between 192.204.106.2 and MY.NET.153.191 are to destination port 1214. This port is usually used by the Kazaa file sharing program and the Unix traceroute according to <http://www.sans.org/d/faq-traceroute-luser> has a source UDP port that is between 33434-33523.

I would assume that this signature only takes the Time To Live (TTL) into account and not the source and destination ports. This could be an effort to evade the IDS crafting a packet so that it expires in front of the IDS causing the Analyst to waste his/her time investigating this event.

The following is a traceroute signature from Snort describing an ICMP traceroute. Here you can see that it generates an event if the TTL is set to 1.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP traceroute "; ttl:1; itype:8; reference:arachnids,118; classtype:attempted-recon;
sid:385; rev:2;)
```

From the information in the alert we are unable to tell if this is UDP or TCP traffic. Since destination port of 1214 is used for Kazaa there is no reason for this traffic from an external source should make it into the clients network.

Recommendations: Implement effective perimeter defense so that this type of traffic does not make it to the clients' internal network.

SCAN Synscan Portscan ID 19104

Top 10 Destination Ports

count	destport
25	1214
3	4662
2	6346
1	23
1	113
1	80

Total number of Rows: 6

All the sources are from external IP addresses and the majority are to ports 1214 and 4662.

```
$ cat alert.020??? | grep "SCAN Synscan Portscan ID 19104" | cut -d "]" -f 3- | cut -d " " -f 2
```

All External Address with only one instance of each source.

These ports are Kazaa and edonkey <http://www.seifried.org/security/ports/4000/4662.html> are both peer to peer file sharing applications used to distribute various digital media across the Internet.

```
03/31-00:04:03.853484 [**] SCAN Synscan Portscan ID 19104 [**] 62.45.42.119:1327 -> MY.NET.150.113:1214
```

```
03/31-06:34:25.976664 [**] SCAN Synscan Portscan ID 19104 [**] 203.62.227.236:3630 -> MY.NET.150.133:1214
```

Again why would you open your internal work to this type of "high risk" traffic.

Recommendations: Implement effective perimeter defense so that this traffic does not actually make it into the internal network.

EXPLOIT x86 setuid 0

Top 10 Source Addresses

count	souceip
7	129.2.146.13
2	163.29.165.236
2	141.213.216.37
2	129.15.133.157
1	203.241.224.41
1	130.91.234.186
1	65.94.217.29
1	200.158.204.117
1	216.122.166.231
1	128.32.165.150

Total number of Rows: 10

*"This alert was generated by an attacker sending a setuid(0) system call to the target. This alert is payload dependant and is identified by its signature of hex payload data b017 cd80 as shown in the Snort rule. SUID files are arguably the most potentially damaging files on *NIX systems should they be successfully exploited. Most attacks that result in root compromise are the result of exploiting SUID files. The next most dangerous file type is the SGID files discussed in the next alert. There is a chance of false positives. These arise most often out of users downloading binary files that may contain hex payload that matches the signatures. This can occur with file sharing apps such as Kazaa or Morpheus" source http://www.giac.org/practical/safka_gcia.doc*

Taking a look at these ports some of them are related to Kazaa (1214) and Edonkey (4662) which are a file sharing application, however some of the port pairs below are not related to any well know port. See below.

```
03/31-02:08:42.232293 [**] EXPLOIT x86 setuid 0 [**] 203.241.224.41:4662 -> MY.NET.150.143:4413
03/31-15:47:33.314123 [**] EXPLOIT x86 setuid 0 [**] 130.91.234.186:6699 -> MY.NET.150.246:1241
03/31-20:12:41.554390 [**] EXPLOIT x86 setuid 0 [**] 163.29.165.236:51828 -> MY.NET.150.143:4662
03/31-21:28:25.186283 [**] EXPLOIT x86 setuid 0 [**] 65.94.217.29:6346 -> MY.NET.88.223:2274
04/01-00:07:42.550798 [**] EXPLOIT x86 setuid 0 [**] 163.29.165.236:51828 -> MY.NET.150.143:4662
04/01-01:40:31.381177 [**] EXPLOIT x86 setuid 0 [**] 200.158.204.117:4837 -> MY.NET.150.143:4662
04/01-16:15:52.908310 [**] EXPLOIT x86 setuid 0 [**] 141.213.216.37:1118 -> MY.NET.88.165:1058
04/01-21:23:28.423361 [**] EXPLOIT x86 setuid 0 [**] 216.122.166.231:80 -> MY.NET.150.106:2452
04/01-21:50:35.970701 [**] EXPLOIT x86 setuid 0 [**] 128.32.165.150:1690 -> MY.NET.150.246:5299
```

04/01-22:19:24.590035 [**] EXPLOIT x86 setuid 0 [**] 130.64.147.33:3602 -> MY.NET.153.191:1214

The only other thing I have to go on is a little research into a possible signature for this alert taken from <http://sunsite.securitycentralhq.com/mirrors/security/snort/Files/Current/exploit.rules> which would indicate that this type of traffic is UDP based. The point being that port pairs from both sides of the connection that are in the ephemeral range would indicate the presence of passive FTP for TCP based connections, which this is not.

alert udp \$EXTERNAL_NET any -> \$HOME_NET any (msg:"EXPLOIT x86 setuid 0"; content: "|b017 cd80|";reference:arachnids,436;)

alert udp \$EXTERNAL_NET any -> \$HOME_NET any (msg:"EXPLOIT x86 setgid 0"; content: "|b0b5 cd80|";reference:arachnids,437;)

Unfortunately <http://www.whitehats.com> is no longer available and no more information is available on this alert.

Other people have seen this alert and can be correlated to other student's particles:

www.giac.org/practical/Carlin_Carpenter_GCIA.doc

www.giac.org/practical/Matthew_Fiddler_GCIA.doc

Matthew Fiddler states that this is a buffer overflow attack. If it is not there is certainly enough activity to host MY.NET.150.143 to warrant a full investigation of this host as has appeared multiple times so far in various alerts

Recommendations: This activity should never make it to the internal network, Implement effective perimeter defense. Investigate host MY.NET.150.143 and put it in a watch list.

Russia Dynamo - SANS Flash 28-jul-00

The SANS Institute wrote:

"SANS Flash Report: Trojans Sending More Data To Russia July 28, 2000, 6:20 pm, EDT

This is preliminary information. The GIAC (Global Incident Analysis Center) has received several submissions showing large amounts of data being sent, illegitimately, from Windows 98 machines to a Russian IP address (194.87.6.X). The cause is most probably a Trojan, but whatever it is, it is moving fast."

Below is an extract of the logs:

03/31-11:43:39.767240 [**] Russia Dynamo - SANS Flash 28-jul-00 [**] 194.87.6.19:3448 -> MY.NET.150.133:1214

03/31-11:43:39.767444 [**] Russia Dynamo - SANS Flash 28-jul-00 [**] MY.NET.150.133:1214 -> 194.87.6.19:3448

03/31-11:43:40.552210 [**] Russia Dynamo - SANS Flash 28-jul-00 [**] 194.87.6.19:3448 -> MY.NET.150.133:1214

This snippet from the logs seems to be Kazaa traffic using port 1214

Recommendations: This traffic appears to be Kazaa traffic. This type of traffic serves no other use than to share files with other users of the Kazaa network. Effective perimeter defense should be implemented to prevent this from happening.

ICMP Destination Unreachable (Protocol Unreachable)

A Packet sent with a protocol value, which is not a valid protocol number would get a response of "ICMP Destination Protocol Unreachable from the destination host. Normally only AIX, HP-UX, Digital Unix machines should send this reply. This would normally be part of someone trying to probe your network as to what type of host may be on the your network. More information on ICMP Usage in scanning can be found here

http://qb0x.net/papers/Scans/ICMP_Scanning_v2.5/

This signature below probably looks similar to the one that generated this alert:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Destination Unreachable (Protocol Unreachable)"; itype: 3; icode: 2; classtype:misc-activity; sid:404; rev:4;)
```

Below is an Extract from the logs:

```
04/08-17:09:10.294352 [**] ICMP Destination Unreachable (Protocol Unreachable)[**] MY.NET.152.17 -> MY.NET.115.133
04/08-17:09:20.345320 [**] ICMP Destination Unreachable (Protocol Unreachable)[**] MY.NET.152.17 -> MY.NET.115.133
04/08-17:51:08.849048 [**] ICMP Destination Unreachable (Protocol Unreachable)[**] MY.NET.153.182 -> 167.206.193.76
```

The Following hosts sent the offending packets to illicit this response:

```
167.206.193.76
24.200.165.198
MY.NET.115.133
MY.NET.221.54
```

The two internal host have not been involved in any other activity other than this event so its not actually known why this was trigger from an internal hosts unless it is some sort of experimental protocol or load balancing protocol. A packet capture would provide more information.

Recommendations: Use of ICMP should be restricted for only internal hosts. The two internal hosts should be check if that they are not part of some sort pf load balancing cluster or used in protocol experimentation.

SUNRPC highport access!

Sun RPC is a port mapper service, that is used to find which ports other services may be listen on

The signature may have looked like the one below:

```
alert tcp any any -> $HOME_NET 32771 (msg: "SUNRPC highport access!";)
```

Below is an extract from the logs:

```
04/04-16:46:23.851682 [**] SUNRPC highport access! [**] MY.NET.6.39:143 -> MY.NET.88.130:32771
<snip>
04/04-21:39:03.093513 [**] SUNRPC highport access! [**] MY.NET.6.39:143 -> MY.NET.88.130:32771
04/05-13:54:36.446182 [**] SUNRPC highport access! [**] MY.NET.253.114:80 -> MY.NET.88.130:32771
04/05-13:54:36.449226 [**] SUNRPC highport access! [**] MY.NET.253.114:80 -> MY.NET.88.130:32771
04/05-13:54:36.450711 [**] SUNRPC highport access! [**] MY.NET.253.114:80 -> MY.NET.88.130:32771
04/05-13:54:36.452013 [**] SUNRPC highport access! [**] MY.NET.253.114:80 -> MY.NET.88.130:32771
04/08-12:27:19.914856 [**] SUNRPC highport access! [**] 131.118.254.38:80 -> MY.NET.88.130:32771
04/08-12:29:19.707391 [**] SUNRPC highport access! [**] 131.118.254.38:80 -> MY.NET.88.130:32771
04/08-12:29:21.556965 [**] SUNRPC highport access! [**] 131.118.254.38:80 -> MY.NET.88.130:32771
```

What I believe here is that we are seeing some false positives and what we are actually seeing is some web server activity and some IMAP. Its just the client used the 32771 as source port when it set up the connection

Correlations: Other SANS students have seen this behavior

http://www.giac.org/practical/Robert_Neel.doc

http://www.giac.org/practical/Raja_Azrina_Raja_Othman.doc

Recommendations: If there are no servers running SUN RPC than disable this rule or it could be rewritten to take only trigger when connections are coming from external networks. Either way this rule will produce a number of false positives

Port 55850 tcp - Possible myserver activity - ref. 010313-1 & Port 55850 udp - Possible myserver activity - ref. 010313-1

This alerts is to detect the possible presence of the MyServer DDos agent. This has been noted seen in the wild and reported on some mailing lists see <http://lists.insecure.org/incidents/2000/Aug/0228.html> and <http://www.sans.org/y2k/082200.htm>

Below is an extract from the logs:

04/03-10:18:01.477425 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**] 200.207.47.117:55850 -> MY.NET.150.133:1214

04/03-10:18:01.477768 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**] MY.NET.150.133:1214 -> 200.207.47.117:55850

04/07-15:34:32.894711 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**] MY.NET.88.189:55850 -> 63.70.44.83:80

While the activity is does include port 55850, in the logs that we have access to all have other legitimate ports making up the socket pair. That being Port 80 and 1214. The appearance of 55850 is completely legitimate as a source port in a TCP connection.

Correlations: This activity has been seen in the wild see the above URL's

Recommendations: As the alert is generated from traffic in either direction, It would be better if the it was made to only take into a destination port of 55850

Back Orifice

Top 10 Source Addresses

count	souceip
6	MY.NET.6.48
5	MY.NET.6.49
4	MY.NET.6.52
2	MY.NET.6.50
1	MY.NET.152.173
1	66.28.14.37
1	MY.NET.6.51

Total number of Rows: 7


```
alert udp any any -> $HOME_NET 31337 (msg:"Back Orifice";)
```

```
alert tcp $HOME_NET 80 -> $EXTERNAL_NET any (msg:"BACKDOOR BackOrifice access"; flags: A+; content: "server|3a| BO|2f|");
```

```
alert udp $EXTERNAL_NET any -> $HOME_NET 31337 (msg:"BACKDOOR BackOrifice access"; content:"|ce63 d1d2 16e7 13cf 39a5 a586|";
```

Back Orifice being a back door you would expect to see kinds of activity which would be either some sort of interactivity or some sort of scanning for this Trojan activity.

04/01-14:32:02.469663 [**] Back Orifice [**] MY.NET.6.48:25193 -> MY.NET.152.182:31337

04/01-21:43:45.575116 [**] Back Orifice [**] MY.NET.6.48:123 -> MY.NET.153.206:31337

04/01-22:56:33.452439 **[**]** Back Orifice **[**]** MY.NET.6.48:49099 -> MY.NET.152.250:313

g at the Volume coming from the Top 10 Source I would not consider this trolling Trojans. How

04/03-11:08:20.981405 [**] spp_portscan: portscan

04/03-11:08:23.093789 [**] spp_portscan: End of portscan from 66.28.14.37: TOTAL time(211s) hosts(1) TCP(0) UDP(217) [**]

04/03-11:01:19.076639 **[**] ICMP Fragment Reassembly Time Exceeded [**]** MY.NET.151.95 -> 66.28.14.37

04/03-11:01:20.111477 [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.151.95 -> 66.28.14.37

ngo we have winner, 66.28.14.37 is responsible for more alerts than just the Back Orifice Alert. In fact all of the

the last 3 are in the over all Top 10 scanners.

WEB-MISC http directory traversal

See: <http://www.whitehats.com/IDS/298> and <http://www.whitehats.com/IDS/297>.

Below are the signatures that may have caused this event:

alert tcp \$EXTERNAL_NET any -> \$HTTP_SERVERS 80 (msg:"WEB-MISC http directory traversal"; flags:A+; content: "..\\"; reference:arachnids,298; classtype:attempted-recon; sid:1112; rev:2;)

alert tcp \$EXTERNAL_NET any -> \$HTTP_SERVERS 80 (msg:"WEB-MISC http directory traversal"; flags:A+; content: "../"; reference:arachnids,297; classtype:attempted-recon; sid:1113; rev:2;)

Below is an extract from the log:

```
04/01-00:06:34.729596  [**] WEB-MISC http directory traversal [**] 68.49.32.46:19350 -> MY.NET.5.96:80
04/01-00:07:31.580066  [**] WEB-MISC http directory traversal [**] 151.196.170.156:64457 -> MY.NET.5.96:80
04/01-00:16:38.264708  [**] WEB-MISC http directory traversal [**] 151.196.170.156:64476 -> MY.NET.5.96:80
04/01-08:26:16.130488  [**] WEB-MISC http directory traversal [**] 192.233.52.163:1342 -> MY.NET.5.96:80
```

Recommendations: Make sure that all web servers are running the latest security patches and no vulnerable cgi scripts. This activity is common amongst script kiddies. Also make sure that there is effective perimeter defense isolating the non-public web servers from the public internet.

ICMP Echo Request BSDtype

Top 10 Source Addresses

count	souceip
-------	---------

9	MY.NET.60.151
3	MY.NET.6.7

Total number of Rows: 2

This is an ICMP echo request originating from a BSD type Unix machine. A possible signature for this would look like the signature below which was taken from <http://sunsite.securitycentralhq.com/mirrors/security/snort/Files/Current/info.rules>.

```
alert icmp any any -> any any (msg:"ICMP Echo Request BSDtype"; itype:8; content:"|08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17|"; depth:32; reference:arachnids,152;)
```

The alert was also seen in the following practicles

http://www.giac.org/practical/David_Jenkins_GCIA.doc

http://www.giac.org/practical/Joe_Ellis_GCIA.doc

Some port scanning activity was also observed for MY.NET.6.7 but could be put lower on the Priority list since these connections are a low number of connection attempts to one or two host and are more like network error more than anything.

```
04/01-15:24:08.725778  [**] spp_portscan: portscan status from MY.NET.6.7: 13 connections across 1 hosts: TCP(13), UDP(0) [**]
```

```
04/01-15:24:10.746623  [**] spp_portscan: End of portscan from MY.NET.6.7: TOTAL time(5s) hosts(1) TCP(13) UDP(0) [**]
```

Recommendations: This looks like normal network chatter and would low on the priority list of investigation compared to most of the other alerts

RPC tcp traffic contains bin_sh

This could be a sign of an attacker trying to exploit a vulnerability in an RPC service, /bin/sh was seen in the traffic flow which is often seen in these sort of attacks.

More information is available at <http://www.whitehats.com/IDS/545> and <http://www.whitehats.com/IDS/544>.

This is what the signature may have looked like:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 32771: (msg:"RPC tcp traffic contains bin_sh"; flags: A+; content: "/bin/sh"; reference: arachnids,545;)
```

Below is an extract of the logs:

```
04/01-16:38:56.023574  [**] RPC tcp traffic contains bin_sh [**] 65.57.83.15:80-> MY.NET.88.189:49996
04/04-17:30:34.399479  [**] RPC tcp traffic contains bin_sh [**] 65.214.56.74:80 -> MY.NET.88.130:32912
04/08-17:42:14.285903  [**] RPC tcp traffic contains bin_sh [**] 65.57.83.15:80-> MY.NET.88.189:49307
04/08-17:42:14.287216  [**] RPC tcp traffic contains bin_sh [**] 65.57.83.15:80-> MY.NET.88.189:49307
```

Correlations: Other People have seen this type of activity

http://www.giac.org/practical/Edward_Peck_GCIA.doc

http://www.giac.org/practical/Joe_Ellis_GCIA.doc

This traffic is probably from an older version of the rule and you can see that this traffic is not port 32771. This would generate a high amount of false positives. In normal traffic. Although I would not expect to see it in web server traffic very often, unless it is a website about shell scripting. As it turns out there is no other suspicious traffic coming from the external IP addresses 65.57.83.15 and 65.214.56.74, the only external addresses for this alert.

Recommendations: Rewrite the rule or snort.conf variables so that it takes the direction into account to reduce the amount of false positives. This alert may also be indicative of another service listening on port 80, eg a telnet daemon listening on port 80 and you may see /bin/sh in the prompt. If a Packet logger were available you be able to see the whole transaction.

EXPLOIT x86 setgid 0

This alert is similar to the "EXPLOIT x86 setuid 0". A setgid(0) system call is sent to the target host. This can result in a root compromise. This is also has a chance for false positives when binary file transfers occur. More information is available at <http://www.whitehats.com/IDS/284>.

The signature may have looked like the following:

```
alert TCP $EXTERNAL any -> $ HOME_NET any (msg: "EXPLOIT x86 setgid 0"; flags: A+; content: "|b0b5 cd80|"; classtype: system-attempt; reference: arachnids,284;)
```

Here you can see most of the connections to ports used by Kaaza and eDonkey which is used for the sharing files across the internet.

```
$ cat alert.020??? | grep "EXPLOIT x86 setgid 0"
```

```
03/31-16:31:40.707909  [**] EXPLOIT x86 setgid 0 [**] 130.91.234.186:6699 -> MY.NET.150.246:1241
03/31-23:37:55.201129  [**] EXPLOIT x86 setgid 0 [**] 211.51.23.101:45021 -> MY.NET.150.143:4662
04/01-10:39:30.676256  [**] EXPLOIT x86 setgid 0 [**] 128.205.181.27:4623 -> MY.NET.88.162:1214
04/03-20:55:11.119425  [**] EXPLOIT x86 setgid 0 [**] 128.248.82.118:1214 -> MY.NET.153.191:2211
04/08-21:01:21.011357  [**] EXPLOIT x86 setgid 0 [**] 131.96.29.152:1214 -> MY.NET.150.113:2882
```

04/08-21:24:04.688687 [**] EXPLOIT x86 setgid 0 [**] 128.12.52.62:1214 -> MY.NET.88.162:1153

This traffic is more likely to be a false positive caused by binary file transfers

This type of activity has been observed in the following GIAC practical:

http://www.giac.org/practical/safka_gcia.doc

Recommendations: This traffic should not even enter the internal network. Implement effective perimeter defense to prevent this type of traffic.

Tiny Fragments - Possible Hostile Activity

Fragmented traffic rarely happens and should be treated with suspicion at all times. It usually means that someone is trying to elude an IDS system or firewall by fragmenting the traffic

The signature could have been similar to the following:

alert ip \$EXTERNAL_NET any -> \$HOME_NET any (msg:"MISC Tiny Fragments"; fragbits:M; dsize: < 25; classtype:bad-unknown; sid:522; rev:1;)

Below is an extract from the logs:

03/31-15:26:23.004569 [**] Tiny Fragments - Possible Hostile Activity [**] 68.56.85.72 -> MY.NET.88.194

03/31-15:26:26.005764 [**] Tiny Fragments - Possible Hostile Activity [**] 68.56.85.72 -> MY.NET.88.194

03/31-15:26:28.744218 [**] Tiny Fragments - Possible Hostile Activity [**] 68.56.85.72 -> MY.NET.88.194

03/31-15:26:34.734271 [**] Tiny Fragments - Possible Hostile Activity [**] 68.56.85.72 -> MY.NET.88.194

All fragmented traffic is from 68.56.85.72, without a packet dump from this host is impossible to verify what was actually going with the fragmentation seen. 68.56.85.72 was not seen in any other hostile activity.

Recommendations: Put 68.56.85.72 on a watch list and see if any other activity comes from that host. Also implement a firewall that can handle fragmentation reassembly.

SCAN FIN

Top 10 Source Addresses

count	souceip
4	64.85.237.71
2	209.176.66.227
2	142.51.44.123
1	80.131.73.50

Total number of Rows: 4

A Fin scan happens when a TCP packet is sent to a destination host with only the FIN flag set. This serves two purposes, one is slip past a firewall that incorrectly handles packets which have an illegal combination of flags set. The second is to illicit a response from an operating system that may identify it so the attacker can find an operating systems specific attack.

If we look at the host 64.85.237.71 in the scan logs we see that this host is specifically targeting host MY.NET.88.162. He/she has probably found this host in an earlier reconnaissance effort is now homing in on the system to find a suitable exploit.

```
$ cat scans.020??? | grep 64.85.237.71
```

```
Apr 8 21:42:14 64.85.237.71:1132 -> MY.NET.88.162:3140 INVALIDACK *2UAPRS* RESERVEDBITS
```

```
Apr 8 21:42:26 64.85.237.71:2591 -> MY.NET.88.162:1214 SYN *****S*
```

```
Apr 8 21:42:26 64.85.237.71:1031 -> MY.NET.88.162:53 FIN *****F
```

```
Apr 8 21:42:29 64.85.237.71:1612 -> MY.NET.88.162:80 SYN *****S*
```

```
Apr 8 21:43:38 64.85.237.71:137 -> MY.NET.88.162:137 FIN *****F
```

The same activity can be observed for the other source hosts in this list.

```
Apr 1 00:52:24 209.176.66.227:514 -> MY.NET.153.191:514 SYNFIN *2****SF RESERVEDBITS
```

```
Apr 1 01:13:51 209.176.66.227:53 -> MY.NET.153.191:3744 FIN *****F
```

```
Apr 4 00:40:51 142.51.44.123:1900 -> MY.NET.88.162:1214 FULLXMAS 1*UAPRSF RESERVEDBITS
```

```
Apr 4 02:16:30 142.51.44.123:2445 -> MY.NET.88.162:1214 XMAS **U*P**F
```

```
Apr 4 02:18:30 142.51.44.123:2445 -> MY.NET.88.162:1214 VECNA *2**P**F RESERVEDBITS
```

This type of scanning technique usually give you an Idea of what the attackers intentions are, here it would be to compromise one of your systems

Recommendations: Put this hosts on a watch list. Investigate the logs again to see what other activities these machines have been up to. Implement a stateful firewall that prevents these type of reconnaissance techniques from ever getting to your internal network.

MYPARTY - Possible My Party infection

Is an email aware worm for win32 platforms. It comes in the form of an email "subject: new photos from my party!" it then installs a trojan and sends a copy of itself to every on in the address book. It the mail it also contains link to www.myparty.yahoo.com

More info can be seen here: <http://www.sophos.com/virusinfo/analyses/w32mypartya.html>.

Below is an extract from the logs:

```
04/03-11:32:31.914193  [**] MYPARTY - Possible My Party infection [**] MY.NET.153.193:1070 -> 209.151.250.170:80
```

```
04/03-11:32:31.914625  [**] MYPARTY - Possible My Party infection [**] MY.NET.153.193:1070 -> 209.151.250.170:80
```

```
04/03-11:32:31.915692  [**] MYPARTY - Possible My Party infection [**] MY.NET.153.193:1070 -> 209.151.250.170:80
```

All logs instances come from MY.NET.153.193, we also see lots of port scanning activity coming come from here as well:

```
spp_portscan: portscan status from MY.NET.153.193: 175 connections across 174 hosts: TCP(4), UDP(171)
```

This type of activity indicates that there is high probability that MY.NET.173.193 has been compromised.

Correlations: Others have seen this activity

http://www.giac.org/practical/Joe_Ellis_GCIA.doc

http://www.giac.org/practical/Bradley_Urwiller_GCIA.pdf

Recommendations: Check MY.NET.153.193 for signs of the Trojan left behind by the "MYPARTY" worm. If this is a positive compromise then remove it from the network to be sanitized.

EXPLOIT x86 stealth noop

Total number of events 7

"EXPLOIT x86 NOOP" signatures are usually written to detect a NOOP sled which precedes a buffer overflow attack. What this means is that a programming parameter that has no bound checking is taken advantage of. This parameter is filled beyond its maximum limit with a series of machine code for "no operation" hence the term NOOP. At the end of the end of the NOOP's a piece of shell code will be given to the machine to execute. As most processes are run with root privileges a system compromise will occur giving the attacker a command prompt with the same privileges as the process that was running. The problem with this signature is that it can generate a lot of false positives. I have seen this many times in some of the networks we monitor, especially for web servers. The signature that generated the alert probably looked like the following:

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"EXPLOIT x86 NOOP"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; depth: 128; reference:arachnids,181; classtype: system-attempt; sid:648; rev:4;)
```

This signature has been revised and should look like this where the more specific destination ports (\$SHELLCODE_PORTS) can be written for this type of attack:

```
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86 NOOP"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; depth: 128; reference:arachnids,181; classtype:shellcode-detect; sid:648; rev:5;)
```

Regardless the signature is still prone to giving false positives and without a payload capture is hard to pass a judgment on what the actual cause was. Now we actually look at the events and see that this is all generated by Web Server traffic:

```
$ cat alert.020??? | grep 207.199.1.201
```

```
04/01-11:16:17.856344  [**] EXPLOIT x86 stealth noop [**] 207.199.1.201:80 -> MY.NET.152.20:2566
04/04-10:33:24.252970  [**] EXPLOIT x86 stealth noop [**] 207.199.1.201:80 -> MY.NET.153.180:2120
04/05-13:51:37.201772  [**] EXPLOIT x86 stealth noop [**] 207.199.1.201:80 -> MY.NET.152.21:1744
04/05-14:35:24.620472  [**] EXPLOIT x86 stealth noop [**] 207.199.1.201:80 -> MY.NET.152.216:4982
04/05-17:04:00.377138  [**] EXPLOIT x86 stealth noop [**] 207.199.1.201:80 -> MY.NET.153.186:1559
04/08-09:07:41.881516  [**] EXPLOIT x86 stealth noop [**] 207.199.1.201:80 -> MY.NET.153.175:1158
04/03-14:48:50.636493  [**] EXPLOIT x86 stealth noop [**] 192.100.104.200:80 ->MY.NET.152.19:3825
```

This type of activity has been observed in following practical.

http://www.giac.org/practical/safka_gcia.doc

Recommendations: See if a packet capture can be taken for this type of activity either by going to the host 207.199.1.201 yourself or setting up a packet logging facility on the Internet access point. I would pass this off as a false positive.

WEB-MISC whisker head

Whisker is a web server scanning tool that will be able to identify what web server you're running, enabling an attacker to greatly improve the chances of success of a successful exploit. It also checks the web server for any of the vulnerable scripts it has in its database. But wait there's more, it also has numerous IDS evasion modes.

More info can be found here <http://sourceforge.net/projects/whisker/> and <http://www.wiretrip.net/rfp/pages/whitepapers/whiskerids.html>

The signature may have looked like:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-MISC whisker HEAD/./*"; flags:A+;  
content:"HEAD/./*";reference:url,www.wiretrip.net/rfp/pages/whitepapers/whiskerids.html; classtype:attempted-recon; sid:1139; rev:4;)
```

Below is an extract from the logs:

```
04/01-10:10:04.135935 [**] WEB-MISC whisker head [**] 12.91.163.139:2162 -> MY.NET.5.96:80  
04/01-10:10:08.857596 [**] WEB-MISC whisker head [**] 12.91.163.139:2167 -> MY.NET.5.96:80  
04/07-19:46:24.295298 [**] WEB-MISC whisker head [**] 12.91.163.151:1249 -> MY.NET.5.96:80
```

The source address are coming from a block allocated to AT&T

OrgName: AT&T WorldNet Services

OrgID: ATTW

NetRange: 12.0.0.0 - 12.255.255.255

CIDR: 12.0.0.0/8

<snip>

Comment: For abuse issues contact abuse@att.net

RegDate: 1983-08-23

Updated: 2002-08-23

TechHandle: DK71-ARIN

TechName: Kostick, Deirdre

TechPhone: +1-919-319-8249

TechEmail: help@ip.att.net

Recommendations: The sys admin should probably get a copy of this tool and run it against all the web servers to see if there are any vulnerabilities. He should also verify that the web servers are configured correctly and are running the latest security patches.

WEB-MISC prefix-get //

There is probably a double slash "//" at the end of the .com, at worst case it may give away what the web server is.

The signature may have looked like this:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-MISC prefix-get //"; flow:to_server,established; uricontent:"get //"; nocase;  
classtype:attempted-recon; sid:1114; rev:4;)
```

The following is an extract from the logs:

```
04/08-13:45:01.790656 [**] WEB-MISC prefix-get // [**] 68.55.250.134:61553 -> MY.NET.5.96:80
```


04/08-13:45:02.344306 [**] WEB-MISC prefix-get // [**] 68.55.250.134:61571 -> MY.NET.5.96:80
04/08-13:45:02.358205 [**] WEB-MISC prefix-get // [**] 68.55.250.134:61572 -> MY.NET.5.96:80

Correlations: Other people have seen this type of traffic

http://www.giac.org/practical/david_stewart_gcia.doc

http://www.giac.org/practical/Scott_Baird_GCIA.doc

Recommendations: Make sure the web servers are patched, if they are exposed to the Internet they will be probed regularly.

WEB-IIS Unauthorized IP Access Attempt

This is a result of an error message generated by a web server when an unauthorized IP address has tried to access it.

This may have been what the signature looked like:

```
alert tcp $HTTP_SERVERS $HTTP_PORTS -> $EXTERNAL_NET any (msg:"WEB-IIS Unauthorized IP Access Attempt"; flow:to_server,established;  
content:"403"; content:"Forbidden\."; classtype:web-application-attack; sid:1045; rev:6;)
```

Below is an extract from the logs:

```
04/08-13:20:07.163616 [**] WEB-IIS Unauthorized IP Access Attempt [**] MY.NET.150.59:80 -> 130.219.157.100:2082  
04/08-13:20:18.781415 [**] WEB-IIS Unauthorized IP Access Attempt [**] MY.NET.150.59:80 -> 130.219.157.100:3709  
04/08-21:22:08.679925 [**] WEB-IIS Unauthorized IP Access Attempt [**] MY.NET.150.59:80 -> 130.227.199.190:1435  
04/08-21:37:01.726131 [**] WEB-IIS Unauthorized IP Access Attempt [**] MY.NET.150.59:80 -> 130.227.199.190:1983  
04/08-21:37:06.840477 [**] WEB-IIS Unauthorized IP Access Attempt [**] MY.NET.150.59:80 -> 130.227.199.190:2192
```

This Net block appears to be registered to:

OrgName: University of Medicine and Dentistry of New Jersey
OrgID: UMDNJ

NetRange: 130.219.0.0 - 130.219.255.255
CIDR: 130.219.0.0/16
NetName: UMDNJ

Recommendations: The web server administrator could answer why this net block is denied access. Other than that it works as designed.

SYN-FIN scan!

The SYN-FIN scan occurs when an attacker *crafts* a packet in the hope it will bypass a firewall or IDS system with the illegal TCP options set. This Being the SYN and the FIN flag. This is so old but people still tend to use. It is only used for reconnaissance and more often than not this might be the first sign of an impending attack, although not from a very skilled attacker.

More info can be found here <http://www.whitehats.com/IDS/198>.

The signature may have looked like this:

```
alert TCP $EXTERNAL any -> $INTERNAL any (msg: "IDS198/scan_SYN FIN Scan"; flags: SF; classtype: info-attempt; reference: arachnids,198;)
```

Below is an extract from the logs:

03/31-09:25:12.785312 [**] SYN-FIN scan! [**] 195.121.234.59:63498 -> MY.NET.153.191:44984
04/07-09:55:36.225150 [**] SYN-FIN scan! [**] 212.211.86.15:23 -> MY.NET.5.79:23
04/07-09:55:36.249808 [**] SYN-FIN scan! [**] 212.211.86.15:23 -> MY.NET.5.25:23
04/07-09:56:42.064051 [**] SYN-FIN scan! [**] 212.211.86.15:23 -> MY.NET.5.79:23
04/08-14:13:54.546402 [**] SYN-FIN scan! [**] 212.211.86.7:23 -> MY.NET.5.79:23

Recommendations: Put 195.121.234.59 on a watch list as it has been involved in other scanning activity such "Null Scans" and implement effective perimeter defense so that none of this traffic makes it to internal hosts.

TFTP - External UDP connection to internal tftp server & TFTP - External UDP connection to internal tftp server

This event occurs when some one is trying to access a TFTP server from the outside, this would be considered serious as the usually the only things that use TFTP are routers to store their config's. Someone having access to these can effectively shut down your whole network or worse.

Below is an extract from the logs:

04/01-12:04:08.755839 [**] TFTP - External UDP connection to internal tftp server [**] 63.250.205.10:256 -> MY.NET.153.46:69
04/03-11:21:28.527684 [**] TFTP - External UDP connection to internal tftp server [**] 63.250.205.36:256 -> MY.NET.153.46:69
04/03-13:30:54.909734 [**] TFTP - External UDP connection to internal tftp server [**] 63.250.219.189:16495 -> MY.NET.153.45:69

Recommendations: Implement effective perimeter defense so that this type of access is not permitted

WEB-CGI formmail access

"This event indicates that a query was made to the formmail CGI program that could allow an attacker to execute arbitrary commands on the server. A vulnerability exists because shell metachars are not properly quoted in the form field parameters". Information taken from <http://www.whitehats.com/IDS/226>

The signature for this event may have looked like the following:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI formmail attempt"; flags:A+; uricontent: "/formmail"; nocase;
content: "%0a"; nocase; reference: bugtraq,1187; reference: cve,cve-1999-0172; reference: arachnids,226; classtype: web-application-attack; sid:1610;
rev:1;)
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI formmail access"; flags:A+; uricontent: "/formmail"; nocase;
reference: bugtraq,1187; reference: cve,cve-1999-0172; reference: arachnids,226; classtype: web-application-activity; sid:884; rev:4;)
```

Below is an extract from the logs:

04/01-07:08:27.826571 [**] WEB-CGI formmail access [**] 65.139.127.189:2600 -> MY.NET.5.95:80
04/04-10:33:09.054465 [**] WEB-CGI formmail access [**] 209.86.205.243:4921 -> MY.NET.150.139:80

Recommendations: Find a better way of sending mail from a web server than using this script.

MISC PCAnywhere Startup

Total Number of Alerts: 3

Below are the 3 related log events:

04/07-20:55:30.875471 [**] MISC PCAnywhere Startup [**] 68.65.112.116:4380 -> MY.NET.151.110:5632

04/08-14:23:49.420251 [**] MISC PCAnywhere Startup [**] 208.228.181.250:5594 -> MY.NET.5.141:5632

04/08-16:54:51.365762 [**] MISC PCAnywhere Startup [**] 208.228.181.250:8419 -> MY.NET.5.141:5632

This indicates a Pcanywhere start session request. PCanywhere is a remote control software for windows, and authorized requests should be investigated.

See <http://www.whitehats.com/IDS/239> for more information.

The signature may have looked like this:

alert udp any any -> any 5632 (msg:"IDS239 - MISC-PCAnywhere Startup"; content:"ST"; depth: "2");)

Recommendations: Implement effective perimeter defense so this traffic does not make it to the internal network.

x86 NOOP - unicode BUFFER OVERFLOW ATTACK

An buffer overflow attack happens when an attacker sends more data to a program then was originally intended for normal use. At the end of this data the attacker inserts extra operations for the target host to execute. This may result in the attacker gaining root privileges on the target host.

The signature may have looked like this:

alert tcp any any -> \$HOME_NET any (msg: " x86 NOOP - unicode BUFFER OVERFLOW ATTACK"; content: "[90009000900090009000]");)

Below is an extract from the logs:

04/03-16:19:00.274064 [**] x86 NOOP - unicode BUFFER OVERFLOW ATTACK [**] 216.117.184.118:80 -> MY.NET.152.248:1267

04/07-17:47:27.284894 [**] x86 NOOP - unicode BUFFER OVERFLOW ATTACK [**] 216.119.102.23:80 -> MY.NET.152.166:2235

You would normally expect the alert to trigger in the opposite direction, here I would say that this event is a false positive and that it was caused by transfer of data from the web server to the client, possible something like flash media content. However with out a packet capture is difficult to take more than an educated guess.

Recommendations: Adjust this rule so that it looks at the external networks rather than any, or live with the false positives, putting effort into investigating each event like this.

suspicious host traffic

There are only two events for this alert:

04/05-19:36:22.265123 [**] suspicious host traffic [**] 12.32.33.178:4072 -> MY.NET.5.44:80

04/07-15:04:47.308735 [**] suspicious host traffic [**] 24.199.229.26:1161 -> MY.NET.5.44:80

Suspicious host traffic I would consider to be a user defined rule similar to a watch list. This was probably made when some suspicious activity was seen coming from or to the host MY.NET.5.44.

Unique activities excluding ports scans:

```
$ cat alert.020??? | grep MY.NET.5.4 | cut -d "[" -f 2 | grep -v spp | sed -e 's/^\[*/g' | sort -u
ICMP Echo Request L3retriever Ping
Possible trojan server activity
SMB Name Wildcard
suspicious host traffic
```

Number of Unique events

```
$ cat alert.020??? | grep MY.NET.5.4 | cut -d "[" -f 2 | grep -v spp | wc | awk '{print $1}'
1604
```

We see that internal host MY.NET.5.4 was flagged as a suspicious host for good reason with over 1604 events and including some possible Trojan activity. 12.32.33.178 and 24.199.229.26 have not been involved in any other events

Recommendations: Find out who owns it and get them to explain the activity, if they can't, perform forensics analysis on the machine

WEB-MISC webdav search access

This is a sign that a remote user has attempted to use the search directive to get list of directories on the web server. This is considered reconnaissance as the attacker could find this information useful for a later attack.

More information can be found at <http://www.whitehats.com/IDS/474>.

The following may be a possible signature for this event:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-MISC webdav search access"; flags:A+; content: "SEARCH "; depth: 8;
nocase; reference:arachnids,474; classtype:web-application-activity; sid:1070; rev:3;)
```

Below is the extract from the log for this event:

```
03/31-01:33:03.186806 [**] WEB-MISC webdav search access [**] 172.167.206.103:1458 -> MY.NET.5.96:80
04/01-00:04:47.708551 [**] WEB-MISC webdav search access [**] 68.49.32.46:19347 -> MY.NET.5.96:80
```

Correlations: Other people have seen this in the wild

http://www.giac.org/practical/Bradley_Urwiller_GCIA.pdf

http://www.giac.org/practical/Joe_Ellis_GCIA.doc

Recommendations: Make sure the web servers have the latest security patches, also remove any unused scripts. If this is a Unix server run the http daemon in jail.

WEB-IIS encoding access

This event may indicate that an attacker has used an invalid hex sequence. This maybe used to bypass access controls on IIS. More information can be found at <http://www.whitehats.com/IDS/200>.

The following may be a possible signature for this event:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS encoding access",flags: A+; content: "|25 31 75|"; reference:arachnids,200; classtype:web-application-activity; sid:1010; rev:3;)
```

Below is the extract from the logs:

```
04/01-00:04:41.135240 [**] WEB-IIS encoding access [**] 68.49.32.46:19347 -> MY.NET.5.96:80
04/01-09:52:38.321954 [**] WEB-IIS encoding access [**] 208.192.129.170:1228 -> MY.NET.5.96:80
```

Correlations: Other people have seen this activity:

http://www.giac.org/practical/Bradley_Urwiller_GCIA.pdf
http://www.giac.org/practical/Carlin_Carpenter_GCIA.doc

Recommendations: Make sure the web servers have the latest security patches, also remove any unused scripts. If this is a Unix server run the http daemon in a jail.

RFB - Possible WinVNC - 010708-1

VNC is a remoter control application, if there is access to this application from an external source it should be investigated with importance.

Below is are the only two events from the logs:

```
04/03-15:45:58.581011 [**] RFB - Possible WinVNC - 010708-1 [**] MY.NET.207.182:5900 -> MY.NET.152.175:4920
04/08-13:56:18.049556 [**] RFB - Possible WinVNC - 010708-1 [**] 66.200.114.146:5900 -> MY.NET.153.196:1860
```

Recommendations: This rule appears to not take into account the traffic flow direction nor does it take into account external and internal networks. Adjust the rule accordingly

Probable NMAP fingerprint attempt

There are only two events for this alert:

```
04/04-04:29:41.012357 [**] Probable NMAP fingerprint attempt [**] MY.NET.253.10:35290 -> MY.NET.150.248:7000
04/04-04:29:41.804927 [**] Probable NMAP fingerprint attempt [**] MY.NET.253.10:35290 -> MY.NET.150.250:7000
```

NMAP is a network mapping tool used what host are available, what services are running and what the possible operating system. It is very configurable and has various evasion techniques to bypass firewalls and Intrusion Detection systems alike.

This would be someone trying to "fingerprint" the operating system so that they can find an appropriate attack from their favorite hacker web site. The interesting thing here is that is from MY.NET.253.10 which is an internal host.

Unique activities excluding portscans:

```
$ cat alert.020??? | grep MY.NET.253.10 | cut -d "[" -f 2 | grep -v spp | sed -e 's/^\[*/g' | sort -u
ICMP Echo Request Nmap or HPING2
NMAP TCP ping!
Null scan!
Probable NMAP fingerprint attempt
```

Number of Unique events

```
cat alert.020??? | grep MY.NET.253.10 | cut -d "[" -f 2 | grep -v spp | wc | awk '{print $1}'  
1100
```

We can see that MY.NET.253.10 has been involved in over 1100 various mapping attempts. If this is not an authorized network auditing station then there is a big problem !

Recommendations: Verify if this is a authorized network auditing station. If not find who owns it and if they can't explain, then its time perform forensics analysis on this machine.

EXPLOIT x86 NOPS

There are only two events for this alert:

```
03/31-17:08:28.327517 [**] EXPLOIT x86 NOPS [**] 211.219.153.49:2243 -> MY.NET.153.164:4971  
04/07-15:25:44.080552 [**] EXPLOIT x86 NOPS [**] 160.79.2.66:0 -> MY.NET.152.244:0
```

Details of this event can be seen at Whitehats www.whitehats.com/IDS/181 . It is characteristics of machine code for no operation or "NOPS". They attacker us trying to pre-pad his buffer overflow with NOPS to increase his chance of success, this is also know as a NOP sled.

The signature may have look like this:

```
alert TCP $EXTERNAL any -> $INTERNAL any (msg: "IDS181/shellcode_shellcode-x86-nops"; flags: A+; content: "|90 90 90 90 90 90 90 90 90 90  
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; classtype: system-attempt; reference: arachnids,181;)
```

It is possible for this sequence to occur in binary file transfer creating a false positive. Without a packet capture it is difficult determine if this is a false positive, however in the second line we see a source an destination port of 0, this does not occur very often and deserves a closer look at the source and destination addresses.

Unique events for 160.79.2.66:

```
$ cat alert.020??? | grep 160.79.2.66 | cut -d "[" -f 2 | grep -v spp | sed -e 's/^\*\*/g' | sort -u  
EXPLOIT NTPDX buffer overflow  
EXPLOIT x86 NOPS  
High port 65535 udp - possible Red Worm - traffic  
ICMP Fragment Reassembly Time Exceeded
```

Unique events for 211.219.153.49

```
$ cat alert.020??? | grep 211.219.153.49 | cut -d "[" -f 2 | grep -v spp | sed -e 's/^\*\*/g' | sort -u  
EXPLOIT x86 NOOP  
EXPLOIT x86 NOPS  
ICMP Fragment Reassembly Time Exceeded
```

Unique events for MY.NET.152.244:

```
$ cat alert.020??? | grep MY.NET.152.244 | cut -d "[" -f 2 | grep -v spp | sed -e 's/^\*\*/g' | sort -u  
EXPLOIT NTPDX buffer overflow  
EXPLOIT x86 NOPS
```

EXPLOIT x86 setuid 0
High port 65535 udp - possible Red Worm - traffic
ICMP Echo Request L3retriever Ping
ICMP Echo Request Nmap or HPING2
ICMP Fragment Reassembly Time Exceeded
INFO Inbound GNUTella Connect request
INFO MSN IM Chat data
INFO Outbound GNUTella Connect request
NMAP TCP ping!
Null scan!
Queso fingerprint
SMB Name Wildcard
Watchlist 000220 IL-ISDNNET-990517
connect to 515 from inside

Unique events for MY.NET.153.164

```
$ cat alert.020??? | grep MY.NET.153.164 | cut -d '"' -f 2 | grep -v spp | sed -e 's/^\*\*/g' | sort -u
```

EXPLOIT x86 NOOP
EXPLOIT x86 NOPS
FTP DoS ftpd globbing
High port 65535 udp - possible Red Worm - traffic
ICMP Echo Request Nmap or HPING2
ICMP Fragment Reassembly Time Exceeded
INFO Inbound GNUTella Connect accept
INFO Inbound GNUTella Connect request
INFO Outbound GNUTella Connect accept
INFO Outbound GNUTella Connect request
MISC Large UDP Packet
NMAP TCP ping!
SMB Name Wildcard
Watchlist 000220 IL-ISDNNET-990517
Watchlist 000222 NET-NCFC
connect to 515 from inside

160.79.2.66 comes from a broadband provider <http://www.intellispace.net> and can be seen here to have been involved in multiple incidents on the clients network.

MY.NET.152.244 has obviously been tagged because of some suspicious activity has observed coming from it in the past as it is part of "Watchlist 000220 IL-ISDNNET-990517". Similar circumstance are seen for the destination host MY.NET.153.164 as it appears on the same watch list as well as Watchlist 000222 NET-NCFC.

Recommendations: Call the technical admin from Intellispace, inform him that host 160.79.2.66 on his network has been involved in malicious behavior.
Contact details are:

TechHandle: IA43-ARIN
TechName: IP Admin, IP
TechPhone: +1-212-536-7968
TechEmail: ipadmin@intellispace.net

Call the abuse contact at kornet in Korea and inform him that host 211.219.153.49 on his network has been involved in malicious behavior.
Contact Details are:

[ISP Network Abuse Contact Information]
Name : Ryu Hyun-Jin
Phone : +82-2-3675-1499
Fax : +82-2-747-8701
E-mail : abuse@kornet.net

Take MY.NET.152.244 and MY.NET.153.164 offline, investigate it and clean it. Investigate what other hosts they have connected to internally and caused IDS events. Implement effective perimeter defense so that it reduces the chances of internal hosts being compromised again and limits the traffic to hardened hosts to provide internet services for the internal hosts.

WEB-IIS asp-dot attempt

There is no a lot of information on this attack, although it is looking at uri content ".asp." .This could be an attacker looking for backup or trying to execute asp scripts that may have been renamed something else.

The following maybe a signature for this event:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS asp-dot attempt";flags: A+; uricontent:".asp."; nocase; classtype:web-application-attack; sid:997; rev:3;)
```

Below is the event from the log:

```
04/03-23:08:11.122179 [**] WEB-IIS asp-dot attempt [**] 66.77.73.236:2677 -> MY.NET.5.95:80
```

Correlations: Other people have seen this attack:

http://www.giac.org/practical/Bradley_Urwiller_GCIA.pdf

Recommendations: Make sure the web servers have the latest security patches, also remove any unused scripts. If this is a Unix server run the http daemon in jail. You can also verify from the web server logs whether what the attacker was trying to do was successful or not by seeing what the error the web server returned.

WEB-CGI redirect access

This event may indicate some one is trying to append stale information to a URL. More information can be found here <http://online.securityfocus.com/bid/1179>.

The following may be a possible signature for this event:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI redirect access"; flags:A+; uricontent:"/redirect"; nocase;
reference:bugtraq,1179; reference:cve,cve-2000-0382; classtype:attempted-recon; sid:895; rev:3;)
```

Below is the event from the log:

```
04/04-22:48:06.868757 [**] WEB-CGI redirect access [**] 152.163.188.37:45092 -> MY.NET.150.83:80
```

152.163.188.37 has also been logged for another event:

```
04/04-22:48:06.868757 [**] spp_http_decode: IIS Unicode attack detected [**] 152.163.188.37:45092 -> MY.NET.150.83:80
```

Correlations: Other people have seen this activity:

<http://archives.neohapsis.com/archives/incidents/2002-03/0006.html>

<http://www.incidents.org/archives/intrusions/msg05670.html>

Recommendations: Make sure the web servers have the latest security patches, also remove any unused scripts. If this is a Unix server run the http daemon in jail. You can also verify from the web server logs whether what the attacker was trying to do was successful or not by seeing what the error the web server returned.

TELNET access

There is only one event for this alert:

```
04/03-11:42:03.097021 [**] TELNET access [**] MY.NET.5.79:23 -> 200.12.60.87:4147
```

The event "TELNET access" is not an exploit as such but would be more a violation of network usage policy. The policy might state that there shall be no clear text protocols to login in to servers. Here we see that the connection is between an internal and external host, which I would consider very bad. This could also be a sign that someone has installed a backdoor on your network and using one of your hosts as a steppingstone into the rest of the network.

```
$ cat alert.020??? | grep MY.NET.5.79 | cut -d "I" -f 2 | grep -v spp | sed -e 's/^\(.*\)/g' | sort -u
```

High port 65535 udp - possible Red Worm - traffic

ICMP Echo Request Nmap or HPING2

IDS552/web-iis_IIS ISAPI Overflow ida nosize

INFO FTP anonymous FTP

NMAP TCP ping!

Port 55850 udp - Possible myserver activity - ref. 010313-1

SNMP public access

SYN-FIN scan!

TELNET access

WEB-MISC Attempt to execute cmd

I would definitely say that this host is being used as a steppingstone if we grep through the raw data logs, even seeing some alerts that my script did manage to extract from the logs.

Recommendations: Further investigation is warranted on host MY.NET.5.79 as it has high volumes of suspicious activity.

TCP SMTP Source Port traffic

There is only one event for this alert:

04/04-19:47:29.982751 [**] TCP SMTP Source Port traffic [**] 209.242.15.114:25 -> MY.NET.152.141:799

TCP SMTP Source Port traffic can only be two things, either the signature is bad and it is picking up a legitimate SMTP connection to a mail server or someone is trying to penetrate an access list on a firewall by using SMTP as the source port, or they are trying to mask their network reconnaissance effort by trying to disguise their scans as normal SMTP traffic.

```
$ cat alert.020??? | grep 209.242.15.114 | cut -d "[" -f 2 | grep -v spp | sed -e 's/^\*\*/g' | sort -u
```

TCP SMTP Source Port traffic

Here it looks like the signature needs to be improved.

Recommendations: Improve the "TCP SMTP Source Port traffic" signature so that it does not cause false positives.

MISC source port 53 to <1024

This event occurs when a traffic from a source port comes from the privileged port range, that being below 1024. This privileged port range is usually used for services running with root privileges.

This may have been the signature:

```
alert tcp $EXTERNAL_NET 53 -> $HOME_NET :1023 (msg:"MISC source port 53 to <1024"; flags:S; reference:arachnids,07; classtype:bad-unknown; sid:504; rev:2;)
```

This is the event that cause this alert:

04/04-17:03:17.772692 [**] MISC source port 53 to <1024 [**] 63.146.181.137:53 -> MY.NET.88.155:0

Here we see a source port of 53, I would be more inclined to see the destination port of 0 more suspicious and that this alert is more a response towards host MY.NET.88.155. After we investigate the destination address for this alert further we see that it actually has been involved in various other activity as you can see below:

EXPLOIT NTPDX buffer overflow [**] 63.146.181.125:123 -> MY.NET.88.155:123

High port 65535 udp - possible Red Worm - traffic [**] 63.146.181.107:65535 -> MY.NET.88.155:65535

<snip>

High port 65535 udp - possible Red Worm - traffic [**] 66.77.13.144:65535 -> MY.NET.88.155:65535

ICMP Echo Request L3retriever Ping [**] MY.NET.88.155 -> MY.NET.5.4

ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.155 -> 63.146.181.129

ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.155 -> 63.146.181.145
ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.155 -> 66.77.13.107
ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.155 -> 66.77.13.110
ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.155 -> 66.77.13.122
MISC source port 53 to <1024 [**] 63.146.181.137:53 -> MY.NET.88.155:0
SMB Name Wildcard [**] MY.NET.88.155:137 -> MY.NET.5.4:137

The source has only been involved in one other incident and that for event "High port 65535 udp - possible Red Worm – traffic"

Recommendations: Take MY.NET.88.155 off the air for further analysis as it appears it has been compromised, also put 63.146.181.137 on a watch list or block its activity from this source at the perimeter and notify the IP space coordinator about hostile activity from this host by using the whois database.

INFO - Web Dir listing

This event occurs when the a web server sends back page with the directory listing of what directories are on the web server. This may be the result of an attacker sending a SEARCH or PROFIND directive to the web server or a web server miss configuration.

The signature for this event may have looked like this:

```
alert tcp $HTTP_SERVERS 80 -> $EXTERNAL_NET any (msg:"INFO - Web Dir listing"; content:"Directory Listing of"; nocase;)
```

Below is the event from the log that was the cause of this alert:

```
04/07-11:46:26.129321 [**] INFO - Web Dir listing [**] MY.NET.150.139:80 -> 149.225.38.252:1090
```

Correlations: Other people have seen this activity:

http://www.giac.org/practical/Mike_Poor_GCIA.doc
http://www.giac.org/practical/John_Jenkinson_GCIA.doc

Recommendations: Make sure the web servers have the latest security patches, also remove any unused scripts. If this is a Unix server run the http daemon in jail. You can also verify from the web server logs whether what the attacker was trying to do was successful or not by seeing what the error the web server returned.

IDS475/web-iis_web-webdav-propfind

This event occurs when a when attacker sends a PROFIND directive to the web server in order to get a directory listing. This may give an attacker useful information in later attack. More information can be found at <http://www.whitehats.com/IDS/475>

The following may be a possible signature:

```
alert TCP $EXTERNAL any -> $INTERNAL 80 (msg: "IDS475/web-iis_web-webdav-propfind"; flags: A+; content: "PROPFIND "; nocase; classtype: info-attempt; reference: arachnids,475;)
```

Below is the event from the log:

```
04/04-11:24:24.478462 [**] IDS475/web-iis_web-webdav-propfind [**] 207.172.11.147:60482 -> MY.NET.5.96:80
```

Correlations: Others have seen this event:

<http://archives.neohapsis.com/archives/incidents/2001-09/0024.html>

http://www.giac.org/practical/Bradley_Urwiller_GCIA.pdf

Recommendations: Make sure the web servers have the latest security patches, also remove any unused scripts. If this is a Unix server run the http daemon in jail. You can also verify from the web server logs whether what the attacker was trying to do was successful or not by seeing what the error the web server returned.

ICMP Destination Unreachable (Host Unreachable)

An ICMP Destination Unreachable (Host Unreachable) is generated by a router who can not resolve the IP address of a host using ARP that is directly connected to it. The error identifies the host that actually sent the packet to the host that could not be resolved. A forged ICMP error message of this type could disrupt traffic flow from the sending host. It could also mean that the destination host is temporarily offline.

This is a possible signature for this alert:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Destination Unreachable (Host Unreachable)"; itype: 3; icode: 1; sid:399;
classtype:misc-activity; rev:4;)
```

This is the event that caused the alert:

```
04/08-16:46:32.946806 [*] ICMP Destination Unreachable (Host Unreachable) [*] MY.NET.157.240 -> MY.NET.88.162
```

Since this is the only event I would probably say that the host is temporarily offline that is sitting behind the router MY.NET.157.240. If it was a routing error or if it was malicious behavior I would expect to see more than one of these events, that also in conjunction with the fact that both sources are internal.

Recommendations: Works as designed!

Top 10 Scanners [\(To Top of Document\)](#)

Count Source Host

784628	MY.NET.60.43
457190	MY.NET.150.143
264500	MY.NET.6.45
253056	MY.NET.6.48
248505	MY.NET.6.49
231280	MY.NET.6.52
198183	MY.NET.6.50
153243	MY.NET.11.8
112244	MY.NET.6.53
97307	MY.NET.6.60

Scanning activity is intelligence gathering, an attacker finds out information about your network looking for the hosts you have and their potential weaknesses. The attacker then will move in for the kill. Scanning activity can shed some light on the disposition of the attacker, whether he or she is an opportunist or is

highly motivated looking for a hole in a particular point in the network. The alarming point here is all the sources are internal. I will attempt shed some light on these activities.

```
$ cat scans.020??? | cut -d " " -f 4- | cut -d ":" -f 1- | egrep -e "^MY.NET.60.43:" | more
MY.NET.60.43:123 -> MY.NET.153.196:2964 UDP
MY.NET.60.43:123 -> MY.NET.153.144:2929 UDP
MY.NET.60.43:123 -> MY.NET.153.173:3407 UDP
MY.NET.60.43:123 -> MY.NET.153.141:3092 UDP
```

MY.NET.60.43 appears to be using the NTP protocol as a source port to map the internal network, the behavior is abnormal because normal NTP behavior uses the same destination and same source port and would only be between a few hosts. An exception to this is if MY.NET.60.43 was the time server for the network, however the source and destination ports would still be 123.

Below is sample of normal NTP behavior.

```
18:10:41.526445 192.168.10.8.ntp > 134.214.100.6.ntp: v4 client strat 2 poll 10 prec -18 (DF) [tos 0x10]
18:10:41.690860 134.214.100.6.ntp > 192.168.10.8.ntp: v4 server strat 2 poll 10 prec -17 (DF)
18:15:00.534438 192.168.10.8.ntp > 193.67.79.202.ntp: v4 client strat 2 poll 10 prec -18 (DF) [tos 0x10]
18:15:00.585130 193.67.79.202.ntp > 192.168.10.8.ntp: v4 server strat 1 poll 10 prec -19
```

Sending UDP traffic to a host where that port is not listening, will result in the target replying with a ICMP Destination Unreachable, Port Unreachable. The attacker would be interested in the response if the intention is to map a network. Using source port 123 would hopefully disguise the attackers activity as NTP.

```
$ cat scans.020??? | cut -d " " -f 4- | cut -d ":" -f 1- | egrep -e "^MY.NET.150.143:" | more
MY.NET.150.143:1057 -> 80.196.17.157:4665 UDP
MY.NET.150.143:1057 -> 217.216.121.127:4665 UDP
MY.NET.150.143:1057 -> 217.85.245.36:4665 UDP
MY.NET.150.143:1057 -> 217.86.27.77:4665 UDP
MY.NET.150.143:1057 -> 61.15.118.79:4665 UDP
MY.NET.150.143:1057 -> 217.83.172.116:4665 UDP
MY.NET.150.143:1057 -> 213.20.24.123:4715 UDP
MY.NET.150.143:1057 -> 217.227.71.57:4665 UDP
```

MY.NET.150.143 appears to be more Kazaa, eDonkey type activity going to thousands of external sites. I would consider this normal behavior for this type of application as that's what it does when searching for files to download.

```
$ cat scans.020??? | cut -d " " -f 4- | cut -d ":" -f 1- | egrep -e "^MY.NET.6.45:" | more
MY.NET.6.45:123 -> MY.NET.153.160:3126 UDP
```

```
MY.NET.6.45:123 -> MY.NET.153.181:3035 UDP
MY.NET.6.45:7000 -> MY.NET.153.173:7001 UDP
MY.NET.6.45:7000 -> MY.NET.153.144:7001 UDP
MY.NET.6.45:7000 -> MY.NET.152.162:7001 UDP
MY.NET.6.45:123 -> MY.NET.153.159:3142 UDP
MY.NET.6.45:123 -> MY.NET.153.142:2958 UDP
```

MY.NET.6.45 apart from exhibiting the same NTP scanning techniques, it appears to be an AFS server, AFS is similar to a Network File System server. I would consider this very bad as obviously an AFS server is a trusted server by other hosts on the networks and if it is compromised it would mean that someone has access to possibly confidential files.

```
$ cat scans.020??? | cut -d " " -f 4- | cut -d ":" -f 1- | egrep -e "^MY.NET.6.48:" | more
MY.NET.6.48:7000 -> MY.NET.153.216:7001 UDP
MY.NET.6.48:516 -> MY.NET.153.216:1552 UDP
MY.NET.6.48:0 -> MY.NET.153.216:0 UDP
MY.NET.6.48:33365 -> MY.NET.153.216:17408 UDP
MY.NET.6.48:29753 -> MY.NET.153.216:4366 UDP
MY.NET.6.48:8488 -> MY.NET.153.216:17886 UDP
MY.NET.6.48:209 -> MY.NET.153.216:65535 UDP
MY.NET.6.48:3056 -> MY.NET.153.216:56711 UDP
MY.NET.6.48:1185 -> MY.NET.153.216:56234 UDP
MY.NET.6.48:36352 -> MY.NET.153.216:26570 UDP
```

MY.NET.6.48 is attempting a UDP portscan on host MY.NET.153.216 with random source ports. It also does the same to a few other hosts. The same activity is observed from MY.NET.6.49, MY.NET.6.52, MY.NET.6.50, MY.NET.6.53 and MY.NET.6.60. This is something I would consider a real problem as these source hosts are internal hosts and they are scanning more internal hosts. This means that they are either network Auditing stations or they have been compromised and are looking for more hosts to add to their army of drones.

```
$ cat scans.020??? | cut -d " " -f 4- | cut -d ":" -f 1- | egrep -e "^MY.NET.11.8:" | more
MY.NET.11.8:1347 -> MY.NET.152.247:1346 UDP
MY.NET.11.8:1347 -> MY.NET.152.10:1346 UDP
MY.NET.11.8:1347 -> MY.NET.152.181:1346 UDP
MY.NET.11.8:1347 -> MY.NET.152.183:1346 UDP
MY.NET.11.8:1347 -> MY.NET.152.12:1346 UDP
MY.NET.11.8:1347 -> MY.NET.152.185:1346 UDP
MY.NET.11.8:1347 -> MY.NET.152.244:1346 UDP
```

MY.NET.11.8:1347 -> MY.NET.152.245:1346 UDP
MY.NET.11.8:1347 -> MY.NET.152.186:1346 UDP

MY.NET.11.8 has a lot of traffic coming from port 1347, this is a multi media conferencing application. This kind of behavior is what I would expect from this type of application.

Top 10 Destinations [\(To Top of Document\)](#)

Count	Destination Address
381249	MY.NET.150.1
318005	MY.NET.151.77
314882	MY.NET.150.198
65281	MY.NET.1.3
59435	MY.NET.150.195
45691	MY.NET.152.1
45127	MY.NET.11.6
34830	MY.NET.11.7
17226	209.10.239.135
13463	MY.NET.11.5

Top 10 Destination Ports [\(To Top of Document\)](#)

Count	Destination Port
637379	515
139678	80
96461	161
90019	137
59349	ICMP
18030	6346
14820	65535
11792	1863
7265	1326
6091	1769

Top 10 Sources [\(To Top of Document\)](#)

Count	Source
-------	--------

317979	MY.NET.150.83
56345	MY.NET.153.118
30561	MY.NET.152.16
29861	MY.NET.152.17
29202	MY.NET.153.126
25223	MY.NET.153.119
20649	MY.NET.11.6
20434	MY.NET.153.164
18950	MY.NET.70.177
18290	MY.NET.152.18

Analysis of the TOP 10 sources can lead to revelations about the nature of the traffic on the network. Recommendations will be given where appropriate. Below is the unique events for each listed IP addresses in the TOP 10 Sources where the host appears as the host address in the connection. That is "source-ip -> destination-ip". What the biggest issue here is all the TOP 10 sources are internal hosts.

Unique events for Source: MY.NET.150.83

Count	Event
-------	-------

317953	connect to 515 from inside
24	SMB Name Wildcard
2	ICMP Destination Unreachable (Protocol Unreachable)

Verify if MY.NET.150.83 is actually host that is running the line printer daemon, this host is most likely a *Nix platform running samba to integrate with a MS environment. You would think this because a "protocol unreachable" message leaving this host would indicate that it is most likely to be a Unix hosts.

Unique events for Source: MY.NET.153.118

Count	Event
-------	-------

56133	connect to 515 from inside
147	spp_http_decod IIS Unicode attack detected
50	INFO MSN IM Chat data
8	ICMP Fragment Reassembly Time Exceeded
4	SMB Name Wildcard

3 ICMP Router Selection

This also looks like a workstation with normal web activity, We can see that there is a lot of Unicode attacks detected this would be more likely be a false positive by bad signatures which don't take traffic flow direction into account. We also see the connect to 515 from inside here for reasons of what is defined as internal networks and what is defines as external networks. It also has been subjected to some sort of spurious fragmentation.

Unique events for Source: MY.NET.152.16

Count	Event
-------	-------

1098	connect to 515 from inside
731	SMB Name Wildcard
728	ICMP Echo Request L3retriever Ping
144	spp_http_decod IIS Unicode attack detected
111	ICMP Echo Request Nmap or HPING2
1	ICMP traceroute
1	High port 65535 udp - possible Red Worm – traffic

MY.NET.152.16 is a little more interesting. Immediately looking at the high volume of "SMB Name wildcard" events you would think that it is a machine on the prowl for scanning for Microsoft machines however all events are only between a couple of machines:

```
MY.NET.152.16:137 -> MY.NET.11.5:137
MY.NET.152.16:137 -> MY.NET.11.6:137
MY.NET.152.16:137 -> MY.NET.11.7:137
MY.NET.152.16:137 -> MY.NET.206.146:137
```

This host has more likely been on the receiving end of a lot of the attacks as some these attacks are responses rather than stimulus as the signatures don't really take direction into account.

If we take a closer look at the event "ICMP Echo Request L3retriever Ping" below, we can see that traffic flowing to the same three destinations which would indicated that MY.NET.11.5, .6 and .7 are NT domain controllers and these are Win2k platforms as per the Whitehats document.

```
$ cat alert.020??? | grep -v spp_port | grep "]" MY.NET.152.16[:]" | grep "ICMP Echo Request L3retriever Ping " | cut -d "]" -f 3- | sort -u
MY.NET.152.16 -> MY.NET.11.5
MY.NET.152.16 -> MY.NET.11.6
MY.NET.152.16 -> MY.NET.11.7
```

The ICMP Echo Request Nmap is a ping packet without payload, this can be a false positives as some network applications can do this. Here again that the amount of alerts seen are due to a poorly configured snort.conf file

Unique events for Source: MY.NET.152.17

Count	Event
-------	-------


```

413 SMB Name Wildcard
405 ICMP Echo Request L3retriever Ping
388 connect to 515 from inside
95     ICMP Echo Request Nmap or HPING2
3     ICMP Destination Unreachable (Protocol Unreachable)
1     spp_http_decod IIS Unicode attack detected

```

Same here due to the lack of what has been defined as internal and external

```

$ cat alert.020??? | grep "ICMP Echo Request L3retriever Ping" | grep "MY.NET.152.17 " | cut -d "]" -f 3- | sort -u
MY.NET.152.17 -> MY.NET.11.5
MY.NET.152.17 -> MY.NET.11.6
MY.NET.152.17 -> MY.NET.11.7
MY.NET.152.17 -> MY.NET.130.166
$ cat alert.020??? | grep "ICMP Echo Request Nmap or HPING2" | grep "MY.NET.152.17 " | cut -d "]" -f 3- | sort -u
MY.NET.152.17 -> MY.NET.11.6
MY.NET.152.17 -> MY.NET.11.7

```

The traffic is the same from as from MY.NET.152.16 and is from the same subnet, this again is starting make more evidence that there is a poorly configured snort environment rather than a compromised machine.

Unique events for Source: MY.NET.153.126

Count	Event
28328	connect to 515 from inside
850	spp_http_decod IIS Unicode attack detected
13	ICMP Router Selection
6	ICMP Fragment Reassembly Time Exceeded
5	SMB Name Wildcard

MY.NET.153.126 is a lot of IIS Unicode attacks detected, probably more than I would consider normal, below is an excerpt and we can see that it was definitely the source. The bottom 10 or so lines look like they had to be generated by a script as the time of the events are so close together and I don't believe that someone can point and click that fast. I believe that there is a problem with this host and should be investigated further.

```

$ cat alert.020??? | grep -v spp_port | grep "]" MY.NET.153.126[:]" | grep "IIS Unicode attack detected" | cut -d "]" -f 1,3- | sort -u
04/01-15:19:43.335964 [**] MY.NET.153.126:2221 -> 18.187.2.150:80
04/01-16:06:38.206470 [**] MY.NET.153.126:3566 -> 18.187.2.150:80
....<snip>.....
04/05-07:45:06.649961 [**] MY.NET.153.126:4875 -> 211.233.30.96:80

```

04/05-07:45:08.891811 [**] MY.NET.153.126:4876 -> 211.32.117.132:80

Unique events for Source: MY.NET.153.119

Count	Event
-------	-------

23126	connect to 515 from inside
2013	spp_http_decod IIS Unicode attack detected
66	INFO MSN IM Chat data
18	ICMP Router Selection

Again as with MY.NET.153.126 we see the same pattern with the IIS Unicode attack. This has definitely got a problem or a problem user and should be investigated further.

```
04/05-15:23:51.477173 [**] MY.NET.153.119:3158 -> 211.233.28.44:80
04/05-15:23:53.385303 [**] MY.NET.153.119:3182 -> 211.233.29.233:80
.....<snip>.....
04/05-15:24:32.987881 [**] MY.NET.153.119:3222 -> 211.233.29.211:80
04/05-15:24:33.021800 [**] MY.NET.153.119:3224 -> 211.233.29.211:80
04/05-15:25:02.518391 [**] MY.NET.153.119:3226 -> 211.233.28.44:80
```

Unique events for Source: MY.NET.11.6

Count	Event
-------	-------

20649	SMB Name Wildcard
-------	-------------------

```
$ cat alert.020??? | grep -v spp_port | grep "]" MY.NET.11.6[:]" | cut -d "]" -f 3 | sed -e 's/:137//g' | sort -u | sort -k 3
MY.NET.11.6 -> MY.NET.152.10
MY.NET.11.6 -> MY.NET.152.11
MY.NET.11.6 -> MY.NET.152.12
MY.NET.11.6 -> MY.NET.152.13
```

This looks like a network configuration/name resolution error of Win32 platforms, all this activity is to the subnet MY.NET.152.X subnet. As this traffic is UDP based and it does not need to set up a connection in order to generate this alert. That in conjunction with 20649 alerts to only 59 unique destination host located in the same subnet, I would most likely say that this is not netbios scanning attempt but rather a network configuration error. The same activity was seen for MY.NET.11.6

Unique events for Source: MY.NET.153.164

Count	Event
19108	connect to 515 from inside
1247	spp_http_decod IIS Unicode attack detected
58	ICMP Fragment Reassembly Time Exceeded
14	INFO Outbound GNUTella Connect request
6	High port 65535 udp - possible Red Worm - traffic
1	INFO Inbound GNUTella Connect accept

It now appears that MY.NET.153.164 as do the other host that have come from that subnet and looks like it has been compromised. As the IIS Unicode attack occurs to frequently and to close together to be a co-incidence, unless there is a major problem with the snort pre processor.

```
04/07-17:50:28.987752 [**] MY.NET.153.164:1914 -> 211.218.150.179:80
04/07-17:51:19.519922 [**] MY.NET.153.164:1938 -> 211.218.150.179:80
04/07-17:51:22.697219 [**] MY.NET.153.164:1944 -> 61.78.61.252:80
04/07-17:51:31.652228 [**] MY.NET.153.164:1947 -> 211.61.51.101:80
04/07-17:51:47.112601 [**] MY.NET.153.164:1953 -> 203.231.231.105:80
04/07-17:51:55.965123 [**] MY.NET.153.164:1960 -> 211.106.67.213:80
04/08-10:26:32.349815 [**] MY.NET.153.164:2119 -> 18.187.2.150:80
04/08-14:45:22.125628 [**] MY.NET.153.164:1674 -> 211.171.249.213:80
```

Unique events for Source: MY.NET.70.177

Count	Event
18902	SNMP public access
32	SMB Name Wildcard
16	Possible trojan server activity

MY.NET.70.177 has an unusually high amount of SNMP activity looking for the public community string. The behavior is also suspicious because also the source ports would normally be 161 or 162 and they are not.

```
$ cat alert.020??? | grep -v spp_port | grep "]" MY.NET.70.177[:]" | grep "SNMP public access" | cut -d "]" -f 3- | sort -u
MY.NET.70.177:1072 -> MY.NET.5.101:161
MY.NET.70.177:1072 -> MY.NET.5.102:161
MY.NET.70.177:1072 -> MY.NET.5.103:161
.....<snip>.....
MY.NET.70.177:1072 -> MY.NET.5.128:161
```

This clearly looks like a scan attempt, maybe an SNMP management station on the internal network looking for SNMP enabled devices. Either way this behavior is concerning and should be verified of indeed this a network management station or not. The Result should then be documented.

Unique events for Source: MY.NET.152.18

Count Event

642	ICMP Echo Request L3retriever Ping
638	SMB Name Wildcard
94	ICMP Echo Request Nmap or HPING2
36	connect to 515 from inside
8	High port 65535 udp - possible Red Worm - traffic

Again this looks like a Win2K host talking to Windows NT domain controllers as the same 3 destinations are seen again

```
$ cat alert.020??? | grep -v spp_port | grep "]" MY.NET.152.18[:|]" | grep "ICMP Echo Request L3retriever Ping" | cut -d "]" -f 3- | sort -u
```

```
MY.NET.152.18 -> MY.NET.11.5
```

```
MY.NET.152.18 -> MY.NET.11.6
```

```
MY.NET.152.18 -> MY.NET.11.7
```

Summary [\(To Top of Document\)](#)

Overall it appears the snort configuration could be improved, defining what is the internal network and what is the external network, as there may be some legitimate traffic that can cause high volumes of traffic like the possible domain controllers MY.NET.11.5, MY.NET.11.6, MY.NET.11.7 and the possible print servers MY.NET.151.77, MY.NET.150.198, MY.NET.150.83.

Effective boundary protection should be implemented so that all traffic that is passing across or into the internal network is screened. All public internet services as the traffic such as web and mail servers should be moved to a screened subnet and should be located on separated interface of a firewall so that no direct connection can be made to the internal network from external networks and also to contain any damaged sustained from an intrusion. The public servers, which MY.NET.5.96 looks like, should be running latest security patches of the services they provide. They should also be check for services they should not be running with tools such as Nmap, Whisker and Nessus. Any scripts on these servers that are not used should be removed. In fact the whole network should be audited on a regular basis.

Users should be re-educated on the acceptable use of network resources, and bandwidth is not consumed for the use of file sharing applications such as Kazaa, eDonkey and others.

Machines that have been recommend for further attention throughout this report should be followed up on.

Appendices [\(To Top of Document\)](#)

How it was done !

Initially I attempted this assignment with some Perl scripts and imported the data into an MySQL database. However while working through the assignment I realized that some of the log files where not in a consistent format and I missed about 40% of the data. So having more confidence in my shell scripts I ran through that data again and did some double checking of the data results and believe that I have more accurate interpretation of the data then what I did initially.

Below is a table of the logs that I used for this assignment. The logs were taken from <http://www.incidents.org/logs/>

Scan Logs	Alert Logs	Out of Spec logs
scans.020331	alert.020331	oos_Apr.1.2002
scans.020401	alert.020401	oos_Apr.2.2002
scans.020402	alert.020403	oos_Apr.3.2002
scans.020403	alert.020404	oos_Apr.4.2002
scans.020404	alert.020405	oos_Apr.5.2002
scans.020405	alert.020408	oos_Apr.6.2002
scans.020406		oos_Apr.7.2002
scans.020408		oos_Apr.8.2002

Appendix 1:

Shell script to extract the data from the Alert logs:

```
#!/bin/sh
#set -x
cat alert.020??? | grep -v spp_portscan | grep \[.*\] | cut -d "[" -f 2 | sed -e 's/\[.*\] //' | sort -u > alertlist.txt

file=alertlist.txt
```

```

# Test if the summary file exist
# if no, create it
# if yes, delete and then recreate it clean
if [ ! -f summary.txt ]
then
    touch summary.txt
else
    rm -f summary.txt
    touch summary.txt
fi

# read the alerts in line by line
# and then count each alert
exec < $file
while read alert
do
    count=`cat alert.020??? | grep "$alert" | wc | awk '{print $2}'`
    echo "$count $alert" >> summary.txt
done

# Sort the summary file in descending order
cat summary.txt | sort -n -r > temp.txt
mv temp.txt summary.txt

# Now get the TOP 10 Destination, Destination Port and source addresses
cat alert.020??? | grep -v spp_portscan | grep \[.*\] | cut -d "[" -f 3- | sed -e 's/\^.*\*] //' | sed -e 's/ -> /:/g' > address.txt
cat address.txt | cut -d ":" -f 1 | sort -u > sourceip.txt
cat address.txt | cut -d ":" -f 3 | sort -u > destip.txt
cat address.txt | cut -d ":" -f 4 | sort -u > destport.txt

if [ ! -f sourcesummary.txt ]
then
    touch sourcesummary.txt
else
    rm -f sourcesummary.txt

```

```

    touch sourcesummary.txt
fi

exec < sourceip.txt
while read source
do
    sourcecount=`cat address.txt | cut -d ":" -f 1 | egrep -e "^$source$" | wc -l`
    echo "$sourcecount    $source" >> sourcesummary.txt
done
cat sourcesummary.txt | sort -n -r | head -n 11 > top10sources.txt

rm -f sourcesummary.txt
touch sourcesummary.txt

exec < destip.txt
while read destination
do
    destcount=`cat address.txt | cut -d ":" -f 3 | egrep -e "^$destination$" | wc -l`
    echo "$destcount    $destination" >> sourcesummary.txt
done
cat sourcesummary.txt | sort -n -r | head -n 11 > top10destinations.txt

rm -f sourcesummary.txt
touch sourcesummary.txt

exec < destport.txt
while read dport
do
    dportcount=`cat address.txt | cut -d ":" -f 4 | egrep -e "^$dport$" | wc -l`
    echo "$dportcount    $dport" >> sourcesummary.txt
done
cat sourcesummary.txt | sort -n -r | head -n 11 > top10destports.txt

echo "Results" > report.txt
echo "-----" >> report.txt
echo >> report.txt

```

```

echo "Summary of Alerts" >> report.txt
echo >> report.txt
echo "Count    Alert" >> report.txt
echo "-----" >> report.txt
cat summary.txt >> report.txt
echo >> report.txt
echo "Top 10 Destinations " >> report.txt
echo "Count    Destination Address" >> report.txt
echo "-----" >> report.txt
cat top10destinations.txt >> report.txt
echo >> report.txt
echo "Top 10 Destination Ports" >> report.txt
echo "Count    Destination Port" >> report.txt
echo "-----" >> report.txt
cat top10destports.txt >> report.txt
echo >> report.txt
echo "Top 10 Source Addresses" >> report.txt
echo "Count    Source Addresses" >> report.txt
echo "-----" >> report.txt
cat top10sources.txt >> report.txt

```

The next script basically just creates the same report as the first one but does some further analysis on the top 10 source ips.

```

#!/bin/sh
#set -x

cat << ENDIT
Summary of Events
Count    Events
-----
ENDIT
cat summaryalerts.txt

echo "Top 10 Destination IPs"
echo "Count    Destination Address"
echo "-----"

```



```

cat summarydestips.txt | head -n 11
echo
echo "Top 10 destination Ports"
echo "Count    Destination Port"
echo "-----"
cat summarydestports.txt | head -n 11
echo
echo "Top 10 Source IPs"
echo "Count      Source"
echo "-----"
cat summarysrcips.txt | head -n 11
echo

SOURCELIST=`cat summarysrcips.txt | head -n 11 | tail -n 10 | awk '{print $2}'`
for SOURCE in $SOURCELIST
do
    cat temp.log | cut -d ":" -f 2,3 | egrep -e ":$SOURCE$" | cut -d ":" -f 1 | sort -u > eventlist.txt
    echo "Unique events for Source: $SOURCE"
    echo "    Count    Event"
    echo "-----"
    echo > uniquesummary.txt
    exec < eventlist.txt
    while read LINE
    do
        COUNT=`cat temp.log | cut -d ":" -f 2,3 | grep "$LINE" | egrep -e ":$SOURCE$" | wc -l`
        echo "$COUNT    $LINE" >> uniquesummary.txt
    done
    cat uniquesummary.txt | sort -n -r
    echo
done

```

The last script is was used to extract the top scanners from the scan logs

```

#!/bin/sh
SCANLOGS=/home/peter/GIAC-Logs

```

```

cat $SCANLOGS/scans.020??? | egrep -e "^w+" | cut -d " " -f 4- | sed -e 's/ -> /:/g' -e 's/ /:/' > scanextract.logs

cat scanextract.logs | sed -e 's/[0-9][0-9]:[0-9][0-9]:[0-9][0-9]:/' > tmp.log
mv tmp.log scanextract.logs

SCANHOSTS=`cat scanextract.logs | cut -d ":" -f 1 | sort -u`
echo > scansummary.txt

for HOST in $SCANHOSTS
do
    COUNT=`cat scanextract.logs | cut -d ":" -f 1 | egrep -e "^$HOST$" | wc -l`
    echo "$COUNT      $HOST" >> scansummary.txt
done

echo "Top 10 Scanners"
echo "Count      Source Host"
echo "-----"

cat scansummary.txt | sort -n -k 1 -r | head -n 10

```

References: [\(To Top of Document\)](#)

- Comnodon Communications, List of known Trojans. Dec 15, 2002
<http://www.comnodon.com/threat/threat-allports.htm>
- The Internet Engineering Task Force, All the RFC's, October 1, 2002
<http://www.ietf.org>
- W. Richard Stevens, TCP/IP Illustrated Volume 1 December 8, 2002
- Stephen Northcutt, Mark Cooper, Matt Fearnow and Karen Frederick Intrusion Signatures and Analysis January 2001. October 20, 2002
- Stephen Northcutt, Judy Novak, Network Intrusion Detection An Analyst's Handbook, 2nd Edition September 2000, October 31, 2002
- Ellie Quigley, Unix Shells by Examples Third Edition 2002 December 10 ,2002
- White Hats Intrusion detection database, Jan 2003
<http://www.whitehats.com>
- SANS, past student practical's, Jan 2003
<http://www.giac.org>
- Secure Networks, Inc. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection January 1998. October 20 2002

<http://secinf.net/info/ids/idspaper/idspaper.html>

- Marty Roesch, Snort an Open Source Network Intrusion Detection System. December 2, 2002

<http://www.snort.org>

- The HoneyNet Project, September 2002

<http://project.honeynet.org/>

© SANS Institute 2003, Author retains full rights.