Global Information Assurance Certification Paper

Rob McBee
SANS Intel, Folsom CA, USA
GIAC GCIA Practical (version 3.3)
Submitted: July 8, 2003

Table of Contents

Assignment #1: Describe the State of Intrusion Detection

ManHunt - Threat Management Defense

Introduction

Today, organizations are faced with the threat of attack from intrusions and denial of service (DoS) attacks. These attacks can come from outside your organization, or from within. Regardless of the type of attack, the recourse available to organizations today is limited. Current security products have, to a certain extent, the capability of keeping attackers out and identifying some threats, but products are needed to actually stop threats and identify the sources of attacks.

Today, an attacker can create a network of thousands of zombie hosts in a matter of hours. New tools using metastasizing and advanced self-replication dispersal techniques can attack trust relationships between systems and build hierarchical levels of zombie networks; making it possible to identify, organize, and control thousands of machines in a matter of minutes.

How it works

ManHunt™ is a protocol analysis-based NIDS. It has the added ability to correlate data taken from other NIDS devices. It works in a peer-to-peer mode for the most part except for its configuration management. ManHunt gives you the ability to identify and analyze threats to your network and proactively respond with the appropriate action.

- Identifying threats that do not necessarily correspond to known attack signatures
- Automating the track-back process within a network and across Internet boundaries to determine the source of the attack, a process that requires significant time and resources when done manually
- Coordinating attack responses automatically, providing faster and more effective attack response
- Correlating events from secondary sources, such as ManTrap, Cisco IDS and Snort devices.
- Providing a detailed log of all activity monitored and responded to by ManHunt so that organizations have the data necessary to take action in response to attacks.

ManHunt is also highly scalable, extensible and flexible: scalable because it performs and coordinates across both large and small networks; extensible because it allows you to monitor multiple devices; flexible because you decide

which devices to monitor, allowing you to focus on problem areas or monitor a broad range of network segments.

Identifying Threats

ManHunt is a new generation of security software that provides the ability to respond to intrusions and prevent damage from denial of service (DoS) attacks. ManHunt monitors the events and performs an analysis to identify the attack. Once an attack is identified, it begins to, "…automatically determining the source of the attack and shares attack information securely with other networks involved in the attack, such as an upstream network service provider, to track the source of the attack across a distributed network...".[1]

ManHunt accomplishes this by using several components that work together to gather event information, analyze the events, and then initiate an appropriate response. This is accomplished by using what ManHunt calls "event sensors". Event sensors gather the information needed to identify the attacks. Event sensors also receive notification of attack events handed off from the other hosts located outside the network. ManHunt samples these events and then performs an analysis that focuses on the devices affected by the attack. Because ManHunt is monitoring event information across the network, it can identify a threat on one part of the network and gather information from additional devices to effectively monitor your network without having to inspect all the traffic.

The two main event sensors incorporated into ManHunt are:

- Sniffer Sensor
- Flow Chaser

Sniffer Sensor

The sniffer sensor allows ManHunt to monitor multiple ports. The sniffer uses switch port analyzers (SPAN) to listen to the network flows that are directly attached to the sniffer by copying all of a port's incoming or outgoing traffic to another port. This enables it to monitor 100% of the traffic on the port without hindering the additional data coming in. When a sniffer detects an attack or another ManHunt hands off information about a suspected attack, the information is passed on to the flow chaser sensor.

Flow Chaser

The flow chaser's job is to query the remote devices to trace the suspicious traffic and verify that an attack is actually taking place. The flow chaser continues querying devices until it finds the attack's entry point into the network.

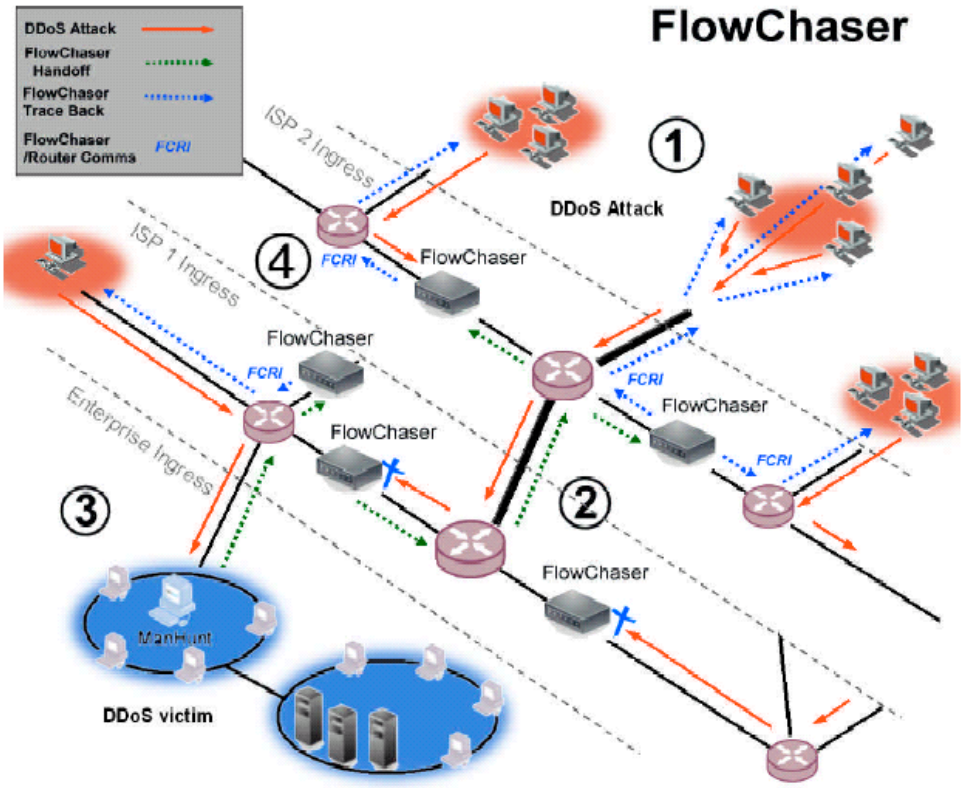


Figure 1: *Diagram of how ManHunt uses FlowChaser to determine the origin of an attack.* [6]

Correlation

To perform the correlation of events, ManHunt uses what it calls the “analysis framework”. Within the analysis framework, ManHunt dynamically updates data it receives from the sensors to the various databases that it can use. ManHunt uses these databases from which it gathers information about attacks, network topology, and local policies. This information, along with data from the sensors, enables ManHunt to determine which action it needs to take in response to a specific attack. Because the databases are constantly being updated, ManHunt has a built-in synchronization service that synchronizes databases across hosts by periodically checking for changes in the local database, then synchronizing the changed database with the databases on the remote hosts that have older versions of the database.

The ManHunt analysis framework aggregates data on a possible attack from multiple sources: other ManHunt hosts, its own event sensors, Cisco IDS, Snort IDS, and ISS RealSecure just to name a few. ManHunt hosts send asynchronous notifications of possible attacks occurring inside or outside the local network. The analysis framework can query the event sensors about its current state and tell it to look elsewhere for more information. The analysis framework further assesses the possible attack by gathering details from its various databases. The image

below was taken from Symantec's web page showing the various sources that can send data events to the ManHunt agents. [2]

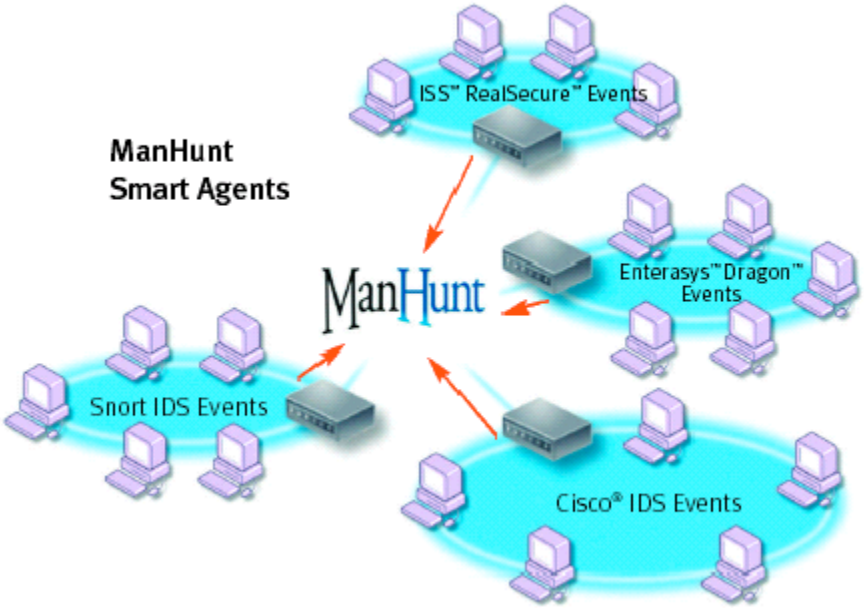


Figure 2: *ManHunt integrates with Third-Party Intrusion Detection Systems to provide Enterprise-Wide Event Correlation and Analysis.* [2]

The analysis framework also performs statistical correlation analysis on events to identify event patterns that vary significantly from usual network activity and to identify individual events that are highly related, such as a port scan followed closely by an intrusion attempt.

The analysis framework schedules allocation of the necessary resources, for example, it might tell the sniffer sensor to monitor traffic on a particular port. The framework then reacts to the data according to the set of rules in the policy database. Depending on the policy, it might begin tracking the attack back to its source or hand off the event to another ManHunt sensor.

Because ManHunt hosts have access to information collected by other ManHunt hosts, attack analysis and response can be coordinated across a distributed network, enabling a rapid global attack response.

Reporting

ManHunt includes multiple levels of reporting to include data drill-down and threat status summaries to quickly spot associated events and attack trends. With these reporting features, you will be able to measure the overall effectiveness of the security infrastructure, meet your operational objectives for measuring security effectiveness, disseminating security information and tracking compliance across the organization.

In addition to the above, ManHunt utilizes a scalable web-based enterprise reporting solution, utilizing the extensive reporting framework of WebTrends™, allowing security administrators to produce powerful and insightful reports to permit them to make better decisions about the threats on their network.

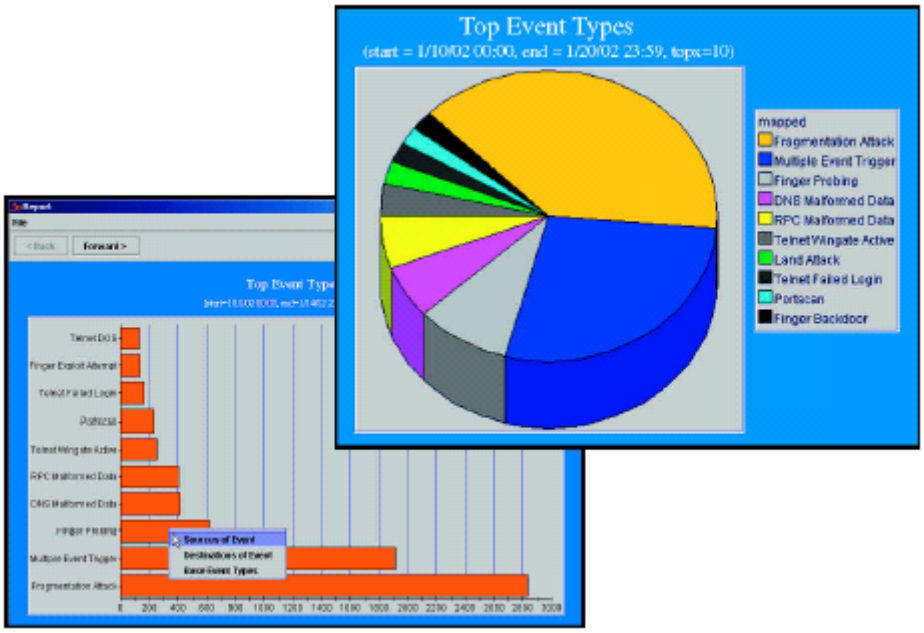


Figure 3: *Reporting options include detailed charts and graphs with drilldown event information that clearly depict attack trends and the overall security posture of the enterprise network.* [2]

Protocol Anomaly Detection

What distinguishes ManHunt from other IDS Systems is this notion of Protocol Anomaly Detection. Unlike most Intrusion Detection Systems today, which are signature based, ManHunt is protocol based. It detects anomalies in the protocol. Protocol anomaly detection is performed at the application protocol layer. Its focus is on the composition of the protocol and looks for misusage of it. ManHunt uses protocol anomaly detection as the heart of its core engine. What ManHunt does is it reconstructs the packets from layer 3 to layer 7 looking for anomalies in the data. If it finds something that does not conform to the RFC standard for that protocol it perceives that as a protocol violation and alerts accordingly. ManHunt supports about 20 supported protocols including the most common ones used in attacks. Some of the supported protocols are Telnet, HTTP, RPC, SMTP, SNMP and Rlogin.

According to Erwan Lemonnier paper from Defcom 2001, he states that, “Protocol anomaly filters are thus able to detect all attacks that are using protocols outside of their normal usage area…which especially includes new

attacks that may not yet have been registered by computer security authorities. This ability of detecting new attacks added to the fact that they don't require signature database updates and has the same long lifetime as the protocol they are monitoring, makes the superiority of protocol anomaly filters on signature filters." [5] He makes a strong point in that protocol anomaly detection can detect new alerts that have not been seen in the wild. By capturing detects early could mean the difference between companies suffering little or no damage to sustaining catastrophe loses. Of course there are limits to this. For instance, if the attack doesn't violate any protocol behavior, then it won't be detected. In this case you would need a signature based IDS to capture the alert.

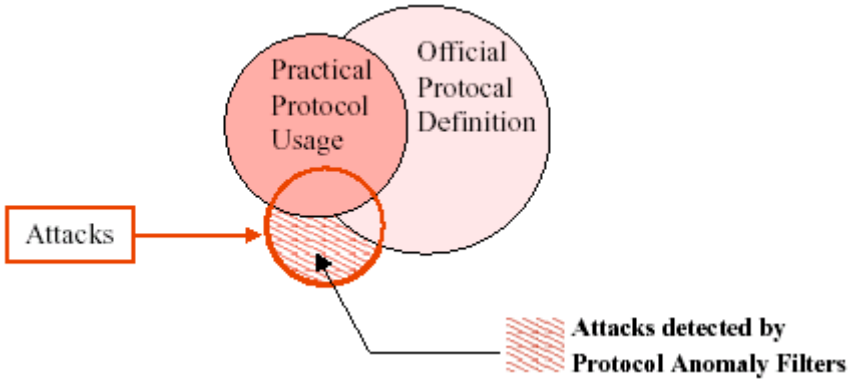


Figure 4: *Attacks detected by Protocol Anomaly Filters.* [5]

False Positives

For optimum incident response you should spend some time getting acquainted with the ManHunt and your network. Before you monitor your network activity, you should finalize an incident response plan. For example, you may want to delegate responsibilities for the incident response team, develop responses for each type of attack, and configure ManHunt to notify designated administrators for specific activity.

To become more acquainted with ManHunt, you should monitor incidents in the administration console and observe your network activity. During this time you will see how your network devices communicate and how the outside world sends and receives data to and from your network.

After you familiarize yourself with ManHunt, you should begin to customize it for your network. Because ManHunt detects all anomalous traffic, you may see events that trigger the ManHunt protocol anomaly detection engine but that are not serious threats to your network. It is highly recommend that you first lower the volume of events to sort through by filtering what you consider to be normal or permitted traffic. Trust me on this one. Once those events are removed, you will have fewer events to analyze; therefore you can allocate your time and energy to the more serious threats.

Now that you've taken the time to reduce the number of events coming into the console, you must now take the appropriate action in responding to them. When examining events, you may determine threatening activity immediately if an intruder breaks into your system or launches a DoS attack against your network. However, many "stealth" attacks occur over long periods of time, so it is important to note suspicious activity and look for patterns over a specified range of time.

You should dissect the event in order to make a subjective judgment based on the incident type, priority, and location. After you dissect the incident, you may decide to examine it further or ignore it. For example, if you see a portscan incident outside of your firewall and your company is highly visible from the Internet, you may decide to ignore this incident as it is probably commonplace. However, if you see the same activity in your high-security financial database subnet, you may want to give the incident more attention. A good place to start is to look for events patterns, related events (or better yet, correlated events), and high-priority events.

What is the Best Approach

What is the best approach to Intrusion Detection? This is a question that is often asked in the security community. In preparation for this paper, I was reading an article on the internet posted by ISP-Planet [3]. On their site, they have an article written by Recourse Technologies [4] that basically says each company should have a layered defense against Intrusion Detection. This is the only way to give your company a fighting chance to the vast array of complex attacks available.

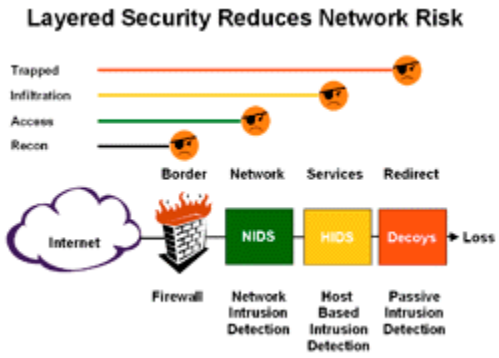


Figure 5: *Layered Security Reduces Network Risk* [3]

Conclusion

As you can see, Protocol Anomaly Detection is much different in the world of Intrusion Detection Systems than say a signature-based solution. To say it would be a better solution, would be a novice statement. It is impossible to find an Off-The-Shelf solution. To be effective in this area of security, one must have a

model made up of several layers to even stand a chance at the various types of threats available today. No one product will be able to detect every attack, but the only way to reduce the impact is to implement multiple layers of security within your organization.

The topic for section one was to describe the state of Intrusion Detection. As you can see, and you probable already know, the rules for Intrusion Detection are ever changing. If you only take one thing from reading this paper, it is that to be effective, you have to implement multiple layers of security. Should ManHunt be one of those layers? It really doesn't matter to me what products you use, just use something. I would consider it because it takes a different approach than just the standard signature based systems. It detects anomalies in the protocol whereas the other big name players do signature matching. Ideally, if you can implement both types, your company stands a better chance of not being owned!

References

1. Earl, Baron. Pigdog Journal. "ManHunt and ManTrap", August 7, 2000. URL: http://www.pigdog.org/auto/scary_tech/link/1628.html (March 10, 2003).

2. Symantec. "ManHunt 2.1", URL: http://enterprisesecurity.symantec.com/content/displaypdf.cfm?pdfid=295&EID=0 (March 11, 2003).

3. Recourse Technologies. IDS-Planet. "Intrusion Detection: Reducing Network Security Risk", December 24, 2001. URL: http://www.isp-planet.com/perspectives/ids_p3.html (March 13, 2003).

4. Recourse Technologies. URL: http://www.recourse.com/index.html (March 13, 2003).

5. Lemonnier, Erwan. Defcon. "Protocol Anomaly Detection in Network-based IDSs" June 28, 2001. URL: http://erwan.lemonnier.free.fr/exjobb/report/protocol_anomaly_detection.pdf (March 16, 2003).

6. Gray, Wayne and Maguire, Andrew. Recourse Technologies. "ManHunt DDoS Attack Trace Back And Prevention Using FlowChaser Technology". URL: Unavailable at this time (March 16, 2003).

7. REACT Network Services, inc. "Products – ManHunt". URL: http://www.reactnetwork.com/manhunt.html (March 19, 2003).

Assignment #2: Three Network Detects

Detect #1: WEB-ISS ISAPI .ida attempt

Trace Log

This is the dump of the alert file of the Snort log that triggered the event:

```
[**] WEB-IIS ISAPI .ida attempt [**]
07/12-14:04:59.354488 66.76.246.193:1343 -> 46.5.180.133:80
TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:1504
***AP*** Seq: 0x486E0A3F  Ack: 0x9225932F  Win: 0x7D78  TcpLen: 20
```

Snort Rule That Triggered the Event

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-IIS ISAPI .ida attempt"; uricontent:".ida?"; nocase;
dsize:>239; flow:to_server,established; reference:arachnids,552;
classtype:web-application-attack; reference:bugtraq,1065;
reference:cve,CAN-2000-0071; sid:1243;  rev:6;)
```

Source of Trace

This trace originated from incidents.org: http://www.incidents.org/logs/Raw/2002.6.12
The file was downloaded in December of 2002 in preparation for the GCIA certification. According to the README file on the incidents.org/logs/Raw page, this trace is the result of a Snort instance running in binary logging mode. The logs have been sanitized and the IP's have been "munged" to protect the guilty.

Detect was Generated By

This trace was generated using Demarc PureSecure v1.6, MySQL 3.23.53 and Snort version 1.9.0-db (Build 209). The trace was downloaded from the incidents.org web site and the following snort command was run against it.

```
snort -p -l C:\PureSecure\sensor\log -r c:\dumps\2002.6.2 –c C:\PureSecure\sensor\conf\snort1.conf
```

The snort1.conf[1] file was the standard file except for the recommendation suggested by Daniel Wesemann on January 5, 2003 to the intrusions@incidents.org mailing list which was to turn off the stream4preprocessor. The following lines were commented out:

```
# preprocessor stream4: detect_scans, disable_evasion_alerts
# preprocessor stream4_reassemble
```

[1] * Normally the snort.conf is not named snort1.conf. This was something Demarc PureSecure did upon install. Not sure why, but everything still worked.

Probability the Source Address was Spoofed

According to Whitehats.com, the packet that caused this event is usually part of an established TCP session indicating that the source IP was not spoofed. However, after going back through ten days worth of logs, there was no associated SYN attempt to start the TCP Handshake. Actually, from those ten days worth of logs, this is the only occurrence of this IP address indicating that the possibility of this address being spoofed exists. An nslookup of this IP shows the following:

> **Name: cdm-66-246-193-tylr.cox-internet.com**
> **Address: 66.76.246.193**

This appears to be a valid host. More than likely this is probably a compromised host. Lets take a closer look at the packet.

Figure 2.1.1: A closer look at this packet

GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN..................x..
%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090
%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a
HTTP/1.0
Content-type: text/xml
HOST:**www.worm.com**

As you can tell from the bolded text, this is obviously a CodeRed request. The www.worm.com shown further down is another sign of CodeRed. Odds are the IP Address is not spoofed. Checking through the logs I did not find any record of the initial SYN attempt, but then I remembered the data present in the logs file was generated by packets that matched a Snort signature. It is highly doubtful that initial SYN would have been logged.

Checking through the logs I found numerous WEB related events on this destination IP. The events included:

- WEB-FRONTPAGE /_vti_bin/ access
- WEB-CGI formmail access
- WEB-MISC whisker HEAD with large datagram
- WEB-IIS cmd.exe access
- WEB-MISC http directory traversal
- WEB-ATTACKS cc command attempt

It definitely looks like active reconnaissance going on among other things.

Description of the Attack

This attack is related to Microsoft IIS Server and is explicitly looking to exploit an unchecked buffer in Microsoft's IIS Index Server ISAPI Extension by requesting non-existent files with .ida or .idq extensions. If the attack is successful, the intruder would gain system level access and have the ability to remotely administer any IIS server running the default install. Having this type of access the attacker would be able to change files and web pages, install applications, delete files, and modify databases among other things. The vulnerability is well documented and a simple Google search will provide you with more information than you know what to do with. CVE-MITRE.org has two listing for this related to IIS and .ida or .idq files.

- CVE-2001-0500
- CVE-2000-0071

This worm has three parts to it:

1. Infect new hosts
2. Deface it's victims Web pages
3. Attack 198.137.240.91 (www1.whitehouse.gov)

IIS Versions susceptible to this attack:

- Microsoft Windows NT 4.0 Internet Information Services 4.0
- Microsoft Windows 2000 Internet Information Services 5.0
- Microsoft Windows XP beta Internet Information Services 6.0 beta

For a more in-depth description check out the article on the SecurityFocus.com web site.

Attack Mechanism

The CodeRed attack starts out by establishing the TCP Handshake. Once the Handshake is completed the attacker will attempt the ISAPI .ida buffer overflow exploit described above in order to gain system level access to the server. If successful, it will proceed to setup the environment and then precede through the remaining steps of the code by initiating 100 other threads, of which 99 check for other vulnerable hosts. The 100th thread will then check to see if the system is running a vulnerable version of IIS, if so, then it will proceed to deface the web page with a page that says, "Welcome to http://www.worm.com !, Hacked By Chinese!". Once the defacing is done, it will check if the c:\notworm file is present. If so, it will go dormant until some specified time period. Then it will check the computer's clock to see if it should send 100k worth of data to port 80 on www.whitehouse.gov essentially performing a DoS attack on www.whitehouse.gov.

Correlations

To confirm my assumptions, I checked through the previous ten days worth of data. While I found no other traces of this source IP, there were multiple instances of other addresses attempting connections to the destination IP, some of which were performing

that same ISAPI .ida buffer overflow attempt. This definitely adds strength to the argument that this in fact is CodeRed. According to the research that Riley Hassel and Ryan Permeh of eEye did, CodeRed's sequence of generating random IP addresses to attack is not exactly random. According to the research, there appears to be a static seed the worm uses when generating IP address to attack. Because of this, each infected host will attempt to infect the same list of addresses. This is precisely what we are seeing in regards to this destination IP.

Another interesting thing I noticed was that this destination IP generated 72 different events of type "ATTACK RESPONSES 403 Forbidden" in response to the stimuli. It was actually 36 unique events, each of which stimulated two responses. Looking at the packets it appears that the initial destination host is not hosting Microsoft IIS but Apache 1.3.12 and it's not even a Microsoft client but a RedHat Linux client. Figure 2.1.2 shows one of the responses.

Figure 2.1.2: Reply from 46.5.180.133

HTTP/1.1 403 Forbidden
Date: Tue, 02 Jul 2002 14:44:52 GMT
Server: **Apache/1.3.12 (Unix) (Red Hat/Linux)** mod_jk mod_ssl/2.6.6 OpenSSL/0.9.5a PHP/4.0.1pl2 mod_perl/1.24 FrontPage/4.0.4.3
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1

Evidence of Active Targeting

I don't believe that is the result of active targeting. The IDS only picked up one event from this source IP. Even though there were multiple other sources that scanned this destination address, this would be expected behavior for CodeRed with its "statically random" IP address algorithm. The other hosts are probably compromised looking for potential victims to infect. It just appears that this destination address is not one of them.

Severity

Severity is calculated using the formula below:

(Criticality + Lethality) - (System Countermeasures + Network Countermeasures)

Each element is worth 1 to 5 points, and the arithmetic gives us a range of -8 to +8.

Criticality	The target in this attack is a public facing web server. Having this server compromised could have catastrophic results to the organization. Even a simple defacement could cost the company significantly.	**4**
Lethality	Had this been a system vulnerable to CodeRed, the attacker would have attempted a buffer overflow resulting in the server being compromised. This would have been very bad.	**5**

System	In this particular case, the target machine does not appear to have any sort of firewall present. Fortunately, this particular host was running Apache as opposed to IIS, so that pretty much eliminates CodeRed infection. Had this been the Slapper worm, this would have been a whole different story. One bad thing is this host is going to generate a lot of noise from all the infection attempts directed towards it.	**4**
Network	From what I can tell there are no network countermeasures in place to prevent this kind of attack from entering target the host's network.	**1**

Severity = (4+5) – (4+1) = **4**

This gives me an incident severity of **4**. As you can see, 4 is probably higher than one would like. There is definitely some work to be done.

Defensive Recommendation

The obvious first choice is something any decent system administrator should do. They should ensure that the systems are patched and cleaned. Microsoft released a patch for this vulnerability some time ago, MS01-033.asp. In addition to just applying this patch, one should also perform Best Known Methods for all systems they administer. Here are a few suggestions:

- Make sure you apply all patches and service packs (after you regression test them of course).
- Harden all your operating systems by making sure you disable all unused user accounts, disabling or removing unneeded services, implement strong passwords, ensure that the proper ACL's are in place whether they are on your routers or on your file systems.
- Keep up-to-date with the latest vulnerabilities by subscribing to the Incidents and Security Focus mailing lists.
- Have a documented security policy that upper management buy's off on so you have the necessary support.
- If for some reason you follow the above steps and still get compromised, try to share with the community so others don't fall victim to the same attack (if your company lets you share this information).

Multiple Choice Test Question

Which of the following Operating Systems are not susceptible to the ISAPI .ida buffer overflow vulnerability? (Choose all that apply)

a) Windows 2000 Advanced Server
b) Windows XP SP1
c) Windows NT
d) Windows 2000 Professional

The answer is (b). Beginning with Windows XP RC1, this vulnerability has been eliminated. (a), (c), (d) are all susceptible to the ISAPI .ida buffer overflow vulnerability.

References

CVE-MITRE.org. “Common Vulnerabilities and Exposures”
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0500 (January 11, 2003).

CVE-MITRE.org. “Common Vulnerabilities and Exposures”
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0071 (January 11, 2003).

Incidents.org. “GIAC Certification Practical Logs”
URL: http://www.incidents.org/logs/Raw/2002.6.12 (December 2002).

Incidents.org. “Intrusions GIAC Mailing List Archives”
URL: http://www.incidents.org/archives (January 5, 2003).

Hassell, Riley, Permeh, Ryan. eEye. “Advisory AD20010618” June 18, 2001.
URL: http://www.eeye.com/html/Research/Advisories/AD20010618.html (January 11, 2003).

Microsoft Corporation. “Unchecked Buffer in Index Server ISAPI Extension Could Enable Web Server Compromise.” June 18, 2001.
URL: http://www.microsoft.com/technet/security/bulletin/MS01-033.asp (January 11, 2003).

SecurityFocus.com. “Code Red Worm” July 20, 2001.
URL: http://aris.securityfocus.com/alerts/codered/010720-Analysis-CodeRed.pdf (January 11, 2003).

Whitehat.com. “arachNIDS IDS552 - IIS ISAPI Overflow IDA”
URL: http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids552&view=event (January 10, 2003).

Detect #2: BACKDOOR Q access

Trace Log

This is the dump of the alert file of the Snort log that triggered the event. For the sake of brevity, only one day is displayed. The event actually triggered for a total of six days starting on 9/28 continuing to 10/3, which is the day shown. In total, 138 events were generated.

```
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
10/03-00:09:38.356507 255.255.255.255:31337 -> 115.74.48.104:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
10/03-00:59:08.336507 255.255.255.255:31337 -> 115.74.74.204:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
10/03-01:31:10.346507 255.255.255.255:31337 -> 115.74.253.150:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
10/03-01:32:07.436507 255.255.255.255:31337 -> 115.74.208.225:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
10/03-04:02:23.366507 255.255.255.255:31337 -> 115.74.142.201:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
10/03-04:56:29.416507 255.255.255.255:31337 -> 115.74.30.152:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
10/03-06:09:01.416507 255.255.255.255:31337 -> 115.74.217.249:515
TCP TTL:16 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
10/03-06:26:00.476507 255.255.255.255:31337 -> 115.74.101.10:515
TCP TTL:16 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Snort Rule That Triggered the Event

alert tcp 255.255.255.0/24 any -> $HOME_NET any
(msg:"BACKDOOR Q access"; flags:A+; dsize: >1;
reference:arachnids,203; sid:184; classtype:misc-activity; rev:3;)

Source of Trace

This trace originated from incidents.org: http://www.incidents.org/logs/Raw/2002.9.3
The file was downloaded in April of 2003 in preparation for the GCIA certification. According to the README file on the incidents.org/logs/Raw page, this trace is the result of a Snort instance running in binary logging mode. The logs have been sanitized and the IP's have been "munged" to protect the guilty. In my quest for more punishment, I imported fourteen days worth of events to hopefully give a more accurate picture of what was going on. The following files were imported:

2002.8.20	**2002.8.21**	**2002.8.22**	**2002.8.23**
2002.8.24	**2002.8.25**	**2002.8.26**	**2002.8.27**
2002.8.28	**2002.8.29**	**2002.8.30**	**2002.9.1**
2002.9.2	**2002.9.3**		

Detect was Generated By

This trace was generated using Demarc PureSecure v1.6, MySQL 3.23.53 and Snort version 1.9.1-db (Build 231). The trace was downloaded from the incidents.org web site and the following snort command was run against it.

snort -p -l C:\PureSecure\sensor\log -r c:\dumps\2002.9.3 –c C:\PureSecure\sensor\conf\snort1.conf

The snort1.conf[2] file was the standard file except for the recommendation suggested by Daniel Wesemann on January 5, 2003 to the intrusions@incidents.org mailing list which was to turn off the stream4preprocessor. The following lines were commented out:

 # preprocessor stream4: detect_scans, disable_evasion_alerts
 # preprocessor stream4_reassemble

Probability the Source Address was Spoofed

The probability the source address was spoofed is definitely high. In this event, as with all the other events, the source address is the broadcast address of 255.255.255.255. According to TCP/IP Illustrated Volume 1 written by W. Richard Stevens,

[2] * Normally the snort.conf is not named snort1.conf. This was something Demarc PureSecure did upon install. Not sure why, but everything still worked.

> "The *limited broadcast address* is 255.255.255.255. This can be used as the destination address of an IP datagram during the host configuration process, when the host might not know its subnet mask or even its IP address. A datagram destined for the limited broadcast address is *never* forwarded by a router under any circumstance. It only appears on the local cable." Page 171.

From this definition we can see that the source address will never be the broadcast address. However, it can be used as the destination address to discover all the devices on your local network. To help strengthen this argument RFC 1122 states that,

> "Limited broadcast. It MUST NOT be used as a source address. A datagram with this destination address will be received by every host on the connected physical network but will not be forwarded outside that network." (Page 28, section c).

Some other things to note are the source port **31337** and destination port **515**. Source port 31337, also known as eleet, is well known for being associated with the Back Orifice Trojan. Typically, Back Orifice events would have this as the destination port and not the source port, but this should definitely raise flags. Even though it is possible that this was a randomly chosen ephemeral port, the odds of all the events choosing this port are slim to none. Destination port 515 is usually associated with the spooler service. I find it odd that a potential outside source would attempt to contact a printer port on an internal network. This is just another raised flag.

Other factors that raised flags to the source address being spoofed and even the packet being crafted are the TTL values. The TTL values ranged anywhere from 14 to 16. While it is feasible that these packets could have made there journey through the internet and arrived with these values, it is highly unlikely. After looking through some of the other events and the packets associated with them, none of them had TTL values even close to this. This is another indication that supports the idea of this packet being crafted. Other factors that also support this theory are the window size being set to 0 and an ID of 0. A window size set to 0 means that the host cannot receive anymore data because its buffer are full and the ID field would usually be set to a random number, that increments by one. These are all factors that strengthen the argument of a crafted packet.

Question posed from the incidents.org mailing list: (This was the only reply I received).

From: msparkz@hotmail.com
Date: Sunday, July 6, 2003 10:32:06 PM
To: rob.mcbee@sbcglobal.net
Subject: RE: LOGS: GIAC GCIA Version 3.3 Practical Detect (mcbee)

Question:
Rob,
Which alert came first? Look at the TTL's, if the order is correct, shouldn't

The TTL be decrementing or is the order reversed?

Answer:
Msparkz is exactly right, in this case I cut a lot of the extra graphs and payloads in order to shrink the size of the file down to a level that incidents.org would post. The final draft that made it through on 7/4/2003 was my third attempt at posting it to the mailing list. Anyway, msparkz would be right if all the packets followed this pattern. When you look at all the events they did not follow this pattern. The only TTL's used for all the events were 14, 15, or 16. This would be indicative of a crafted packet and not the flow of normal path.

The last thing I want to point out, and probably the most obvious, is that each packet has both the ACK and RST flags set in the header. Why would you set the RST flag? The RST flag signals an abrupt termination of the session. This does not make sense.

Below is a summary of the packet with the payload as captured by Demarc PureSecure. Not a whole lot more information, but just another view of the data.

Figure 2.2.1: A closer look at this packet

2002-10-03 07:26:00 SID:1 CID:75321
BACKDOOR Q access
[TCP] 255.255.255.255:31337 -> 115.74.101.10:515
63 6B 6F **cko**

2002-10-03 07:09:01 SID:1 CID:75308
BACKDOOR Q access
[TCP] 255.255.255.255:31337 -> 115.74.217.249:515
63 6B 6F **cko**

2002-10-03 05:56:29 SID:1 CID:75291
BACKDOOR Q access
[TCP] 255.255.255.255:31337 -> 115.74.30.152:515
63 6B 6F **cko**

2002-10-03 05:02:23 SID:1 CID:75282
BACKDOOR Q access
[TCP] 255.255.255.255:31337 -> 115.74.142.201:515
63 6B 6F **cko**

2002-10-03 02:32:07 SID:1 CID:75243
BACKDOOR Q access
[TCP] 255.255.255.255:31337 -> 115.74.208.225:515
63 6B 6F **cko**

2002-10-03 02:31:10 SID:1 CID:75241
BACKDOOR Q access
[TCP] 255.255.255.255:31337 -> 115.74.253.150:515
63 6B 6F **cko**

2002-10-03 01:59:08 SID:1 CID:75229
BACKDOOR Q access

```
[TCP] 255.255.255.255:31337 ->  115.74.74.204:515
63 6B 6F                        cko

_____________________________________________________
2002-10-03 01:09:38   SID:1 CID:75202
BACKDOOR Q access
[TCP] 255.255.255.255:31337 ->  115.74.48.104:515
63 6B 6F                        cko
```

As you can tell from the bolded text, the **63 6B 6F cko** is indicative of the Q signature. This just strengthens the argument that something not right is going on.

Description of the Attack

According to the Honeynet Project, "Q v2.0 is a client / server backdoor which features remote shell access with strong encryption for root and normal users, and a encrypted on-demand tcp relay/bouncer that supports encrypted sessions with normal clients using the included tunneling daemon. Also has stealth features like activation via raw packets, syslog spoofing, and single on-demand sessions with variable ports."

As you can see Q is somewhat sophisticated. To date there are currently three versions of this: Q-1.0, Q-2.0, and Q-2.4. From the documentation available, the main changes/improvements are ease of use and stronger encryption. The latest version, Q-2.4 now uses strong RSA/libiSSL encryption for sessions.

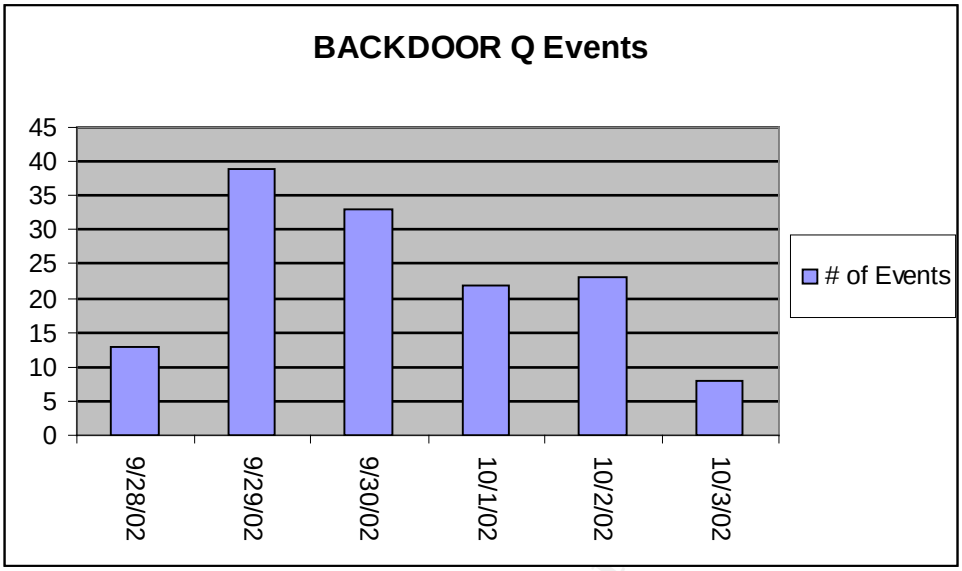
From the Q documentation, in order to use this you have to perform the following:

- Upload "qd" to the system you want to access.
- Type "qd" again to start the server (you must have root privileges to perform this).
- From the client, type q along with you desired parameters (highly craft-able).

From looking at the data, there were 138 attempts made over the six days. The rate at which these events were logged appears to be random with no distinguishable patterns in time. It is definitely too slow to be considered a DoS attack. All attempts were from the broadcast address on source port 31337 and directed towards random targets on the 115.74.0.0/16 network on target port 515. The payloads are all the same with "cko" being the only characteristic. Investigating further shows no other events from the targets being generated. This makes it impossible to distinguish whether this was just a scan of the target or if the target actually initiated the request. My guess is this is an attacker scanning for potentially infected hosts hoping to find some that might have been infected either through email, IRC or maybe through a P2P client.

Below is a graph that shows the number of events per day that were captured matching the BACKDOOR Q access signature.

Figure 2.2.2: Number of Events per day



Attack Mechanism

As we can see from the data presented thus far this traffic is highly questionable. What we can conclude is that no response should be sent back to the source. If for some reason one of the hosts did reply, it should not make it past the perimeter router as this type of data should be blocked. Another interesting factor to consider is that each packet had both the ACK and RST switches set. By setting the RST flag this performs an abrupt termination of the session, making one believe that no response will be sent. This could conceivably be a deliberate attempt at evading an IDS or Firewall; however, what would it prove. After scanning through the logs I didn't find any indication of the three-way handshake. One theory that might be possible is that an attacker is scanning for already infected hosts and if it finds one, the "cko" in the payload details explicit instructions on what to do next…just a theory, but remember the payload may be encrypted.

Correlations

At this point, it's still a mystery as to what is exactly going on. Looking through the fourteen days worth of data I didn't find any indication of three-way handshake being established. At this point I decided to turn to the community. I read a paper written by Les Gordon. He performed a very in-depth analysis of each version. What was interesting to me was his analysis of version Q-2.4. In this version, he says, "qs now allows you to specify the spoofed source IP address of control packets sent to activate servers or execute remote commands. By default, they are randomly generated. The protocol used is now randomly chosen between TCP, UDP and ICMP, unless otherwise specified. This version (and 2.0) seems to have a bug which prevents you from selecting a protocol of your own choosing - you are stuck with the random choice. As well as executing remote commands using the shell predefined in the "conf.h" file, you can now specify what remote program to use as your remote shell using qs."

Figure 2.2.3: qs v2.4 usage message

```
./qs [-p] [-niasd] <-CSB> <host> [more hosts...]
-p <n>          shell/bouncer server listening port
-n              insecure plaintext servers     [encrypted]
-i <n>          protocol (I/U/T)               [random]
-a <n>          custom auth token              [hardcoded]
-s <n>          source IP                      [random]
-S              spawn qshell server
-B <host port>  spawn qbounce to <host port>
-C <cmd>        execute <cmd>
-P <prg>        set a new program as remote shell
-U <uid>        set a new user id for redirecting
```

In his conclusion Les states that, "Q is a potentially very capable back-door program that makes use of several techniques that make its traffic difficult to detect using signature-based IDS." He goes on to say that, "The real danger is perhaps not from Q itself, but from other private software which makes use of similar "stealth" techniques which we don't have signatures for…From this "stealth" technique one could easily craft a packet that could evade an IDS or Firewall".

The one thing that was still bugging me was the "cko" in the payload. Les didn't mention it in his analysis. So I went to Google and did a search on "BACKDOOR Q access" and long behold, numerous links were presented.

1. http://lists.jammed.com/incidents/2001/04/0153.html
2. http://archives.neohapsis.com/archives/incidents/2001-05/0038.html
3. http://maclux-rz.uibk.ac.at/~maillists/focus-ms/msg01549.shtml
4. http://archives.neohapsis.com/archives/incidents/2001-04/0318.html

After reading through these lists, there seems to be a common theme associated with this type of traffic. The theme is IRC sites, Viruses, and Trojans. This leads me to believe that Les was right and there is probably a hacker group that has modified the code to fit their own needs. I do believe the code would have had to of been modified because v2.4 sends an IP Length of 0 where these packets have an IP Length of 20. This is probably intentional to help trick the IDS and Firewall systems. I guess the better question is why. There are still a lot of questions as to why this attacker chose this network. Did they know something about this network or was this just completely random? Unfortunately, with the limited information available one can only guess the answer.

Evidence of Active Targeting

I do believe this could be the result of active targeting. Of the fourteen days that I have inspected, six of them had data present. Plus, the number of events and times differed per day with no noticeable patterns. It's obvious this attacker wasn't trying to be stealthy because they would have picked different source and targets ports instead of some well-known ports like 31337. This one would raise flags all over the place if it wasn't already closed to begin with. It is interesting why no other subnets were scanned or exploited. Maybe the attacker had some knowledge of this subnet because they only made one attempt per host for a total of 138 attempts.

Severity

Severity is calculated using the formula below:

(Criticality + Lethality) - (System Countermeasures + Network Countermeasures)

Each element is worth 1 to 5 points, and the arithmetic gives us a range of -8 to +8.

Criticality	At this point we do have some evidence of active targeting, but there does not appear to be anyone responding to the scans. Plus we have no idea what type of systems are being scanned.	**2**
Lethality	Had one of these systems been susceptible to Q, the attacker would have been granted root access. With this level of access and access to the internal network, there's no telling what mischief they could have caused. Either way this would have been bad.	**4**
System	Unfortunately, we have no way of knowing what kind of protection are in place on the target systems. It does appear that no responses were sent back to the sources so you can safely say that none of them are infected with the Q Trojan.	**2**
Network	One would assume that there some level of filtering in place at the perimeter, but due to the limited knowledge we have on this network nothing is certain. The fact that no responses were seen heading back to the source is good. I had to split down the middle.	**3**

Severity = (2+4) – (2+3) = **1**

This gives me an incident severity of **1**. I have a little work to do, but it's not that big of a deal.

Defensive Recommendation

The obvious first choice is to ensure that ingress and egress filtering is enabled at your perimeter. This will ensure that only permitted traffic is allowed through. You will also want to setup a firewall at your perimeter as well as on your clients. Make sure you patch your clients and servers as this will help reduce the probability of being compromised if someone does get through the perimeter. As part of the hardening process on your network, you could also close down unnecessary ports from the outside that are prone to being exploited.

Multiple Choice Test Question

Given the following packet capture, what is/are the key indicator(s) in determining whether the lpd spooler service is being targeted?

[**] BACKDOOR Q access [**]
10/03-00:09:38.356507 255.255.255.255:31337 -> 115.74.48.104:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
*****A*R** Seq: 0x0 Ack: 0x0 Win: 0x0 TcpLen: 20**

a) The ACK & RST flags are set
b) TTL:15
c) The target port of 515
d) The Seq, Ack, and Win values are all set to 0x0
e) All of the above
f) None of the above

The answer is (c). According to IANA, port 515 is associated with the spooler service. All the other answers have no bearing on what port the source will attempt to connect to.

References

CVE-MITRE.org. "Common Vulnerabilities and Exposures"
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660 (April 24, 2003).

Gordon, Les. Whitehats.ca. "On Q"
URL: http://www.whitehats.ca/main/publications/external_pubs/Q-analysis/Q-analysis.html (April, 25, 2003).

Honeynet Project. "Attacker tools found on apollo.honeyp.edu"
URL: http://project.honeynet.org/challenge/results/submissions/addam/toolkit.txt (April 25, 2003).

Incidents.org. "GIAC Certification Practical Logs"
URL: http://www.incidents.org/logs/Raw/2002.9.3 (April 2002).

Incidents.org. "Intrusions GIAC Mailing List Archives"
URL: http://www.incidents.org/archives (April 24, 2003).

R. Braden, Ed. "RFC1122 - Requirements for Internet Hosts - Communication Layers"
URL: ftp://ftp.rfc-editor.org/in-notes/rfc1122.txt October 1989. (Page 28, section c).

Stevens, W. Richard. "TCP/IP Illustrated, Volume 1" Reading: Addison Wesley, Inc, 1994. (Page 171).

Whitehat.com. "arachNIDS IDS203 - TROJAN-ACTIVE-Q-TCP"
URL: http://www.whitehats.com/info/IDS203 (April 24, 2003).

Detect #3: BAD TRAFFIC bad frag bits

Trace Log

Below is the dump of the alert file of the Snort log that triggered the event.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BAD TRAFFIC bad frag bits [**]
11/10-14:14:53.596507 213.107.68.84 -> 207.166.176.240
TCP TTL:111 TOS:0x0 ID:32146 IpLen:20 DgmLen:1468 DF MF
Frag Offset: 0x0000   Frag Size: 0x0014
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BAD TRAFFIC bad frag bits [**]
11/10-14:14:58.936507 213.107.68.84 -> 207.166.176.240
TCP TTL:111 TOS:0x0 ID:32318 IpLen:20 DgmLen:1468 DF MF
Frag Offset: 0x0000   Frag Size: 0x0014
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Snort Rule That Triggered the Event

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

Source of Trace

This trace originated from incidents.org: http://www.incidents.org/logs/Raw/2002.10.10 The file was downloaded in May of 2003 in preparation for the GCIA certification. According to the README file on the incidents.org/logs/Raw page, this trace is the result of a Snort instance running in binary logging mode. The logs have been sanitized and the IP's have been "munged" to protect the guilty. In my quest for more punishment, I imported ten days worth of events to hopefully give a more accurate picture of what was going on. The following files were imported:

2002.10.1	**2002.10.2**	**2002.10.3**	**2002.10.4**
2002.10.5	**2002.10.6**	**2002.10.7**	**2002.10.8**
2002.10.9	**2002.10.10**		

To help assist my understanding of this detect, I am going to attempt to gain a better understanding of the network. I hadn't done this in the previous two detects, but after reading André Cormier's analysis, I was inspired. Plus, it's more commands that I can learn about.

The first step I attempted was to find the unique MAC addresses for source and target systems. Since I'm not running on a Linux platform, I had to modify things a bit to gain some similar functionality. To accomplish this I installed Cygwin, version 1.3.22-1, along with Windump 3.8 Alpha. Now I have an environment comparable to that of unix. Well, sort of.

The next step is to analyze the file. Below is the sample of file that was generated by typing the following:

C:\Dumps>windump -neqr 2002.10.10 > mac.txt

```
17:24:47.786507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 1482: IP 81.97.214.13.4746 > 207.166.84.248.80: tcp
1428 (frag 36529:1448@0+)
17:25:00.806507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 1482: IP 81.97.214.13.4746 > 207.166.84.248.80: tcp
1428 (frag 37596:1448@0+)
17:25:25.716507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 1482: IP 81.97.214.13.4746 > 207.166.84.248.80: tcp
1428 (frag 39660:1448@0+)
17:26:13.706507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 1482: IP 81.97.214.13.4746 > 207.166.84.248.80: tcp
1428 (frag 43627:1448@0+)
17:27:37.236507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 1482: IP 80.4.97.69.1770 > 207.166.206.194.80: tcp
1416 (frag 43019:1448@0+)
17:27:49.766507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 1482: IP 81.97.214.13.4746 > 207.166.84.248.80: tcp
1428 (frag 50451:1448@0+)
```

As you can see the columns highlighted in blue represent the source and target MAC addresses. The first one is the source, which is in column 2 and the second one is the target, which is in column 3. Below are commands I executed and the results they returned.

```
C:\Dumps>windump -neqr 2002.10.10 | cut -d ' ' -f2 | sort | uniq
0:0:c:4:b2:33
0:3:e3:d9:26:c0

C:\Dumps>windump -neqr 2002.10.10 | cut -d ' ' -f3 | sort | uniq
0:0:c:4:b2:33
0:3:e3:d9:26:c0
```

As you can see there are only two different MAC addresses in use. According to IEEE OUI listing, these two MAC addresses belong to Cisco devices. From this information it is safe to assume that our network looks something like this:

```
CISCO-DEVICE +--+--+ CISCO-DEVICE
                   |
             SNORT INSTANCE
```

Detect was Generated By

This trace was generated using Demarc PureSecure v1.6, MySQL 3.23.53 and Snort version 2.0.0-db (Build 72). The trace was downloaded from the incidents.org web site and the following snort command was run against it.

snort -p -l C:\PureSecure\sensor\log -r c:\dumps\2002.10.10 –c C:\PureSecure\sensor\conf\snort1.conf

The snort1.conf[3] file was the standard file except for the recommendation suggested by Daniel Wesemann on January 5, 2003 to the intrusions@incidents.org mailing list which was to turn off the stream4preprocessor. The following lines were commented out:

 # preprocessor stream4: detect_scans, disable_evasion_alerts
 # preprocessor stream4_reassemble

Probability the Source Address was Spoofed

At first glance of the packet above, you notice that the More Fragments and Don't Fragments bits are set. This automatically tells you that someone is doing some packet crafting. According to RFC 791, setting both fragmentation bits at the same is possible, but it's against the standard. This is not the expected behavior and should be treated as suspicious. Knowing this much information, at first, led me to believe that the source address might be spoofed. But taking a closer look at the payload, see below, showed some remarkable similarities to CodeRed. To confirm my suspicion I checked cert.org and these two packets have the exact same footprint. The one thing that was still puzzling me was I didn't remember CodeRed sending packets with both MF and DF bits set. So, the only conclusion that I can come up with is someone is using a CodeRed packet and crafting it to show both MF and DF bits set. Maybe they were trying to evade an IDS system? Either way this CodeRed footprint is typical of a worm attempting to infect a machine and it's probably looking for some type of reply. If it was a spoofed address, this would be meaningless. Because of this, I would say that the probability of the source address being spoofed is low.

Below is the hex output of the packet generated by windump. The following command was executed to generate this payload.

windump -X -x -vv -r 2002.10.10 "tcp and src 213.107.68.84"

Figure 2.3.1: A closer look at this packet

```
0x0000   4500 05bc 7d92 6000 6f06 189d d56b 4454 E...}.`.o....kDT
0x0010   cfa6 b0f0 12be 0050 2618 748b 2d96 92a6 .......P&.t.-...
0x0020   8018 faf0 d3a6 0000 0101 080a 0002 9d03 ................
0x0030   55d9 fd67 4745 5420 2f64 6566 6175 6c74 U..gGET./default
0x0040   2e69 6461 3f4e 4e4e 4e4e 4e4e 4e4e 4e4e .ida?NNNNNNNNNNNN
0x0050   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e NNNNNNNNNNNNNNNN
0x0060   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e NNNNNNNNNNNNNNNN
0x0070   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e NNNNNNNNNNNNNNNN
0x0080   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e NNNNNNNNNNNNNNNN
0x0090   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e NNNNNNNNNNNNNNNN
0x00a0   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e NNNNNNNNNNNNNNNN
0x00b0   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e NNNNNNNNNNNNNNNN
0x00c0   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e NNNNNNNNNNNNNNNN
0x00d0   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e NNNNNNNNNNNNNNNN
0x00e0   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e NNNNNNNNNNNNNNNN
```

[3] * Normally the snort.conf is not named snort1.conf. This was something Demarc PureSecure did upon install. Not sure why, but everything still worked.

```
0x00f0  4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x0100  4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x0110  4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x0120  4e4e 4e4e 4e25 7539 3039 3025 7536 3835 NNNNN%u9090%u685
0x0130  3825 7563 6264 3325 7537 3830 3125 7539      8%ucbd3%u7801%u9
0x0140  3039 3025 7536 3835 3825 7563 6264 3325      090%u6858%ucbd3%
0x0150  7537 3830 3125 7539 3039 3025 7536 3835      u7801%u9090%u685
0x0160  3825 7563 6264 3325 7537 3830 3125 7539      8%ucbd3%u7801%u9
0x0170  3039 3025 7539 3039 3025 7538 3139 3025      090%u9090%u8190%
0x0180  7530 3063 3325 7530 3030 3325 7538 6230      u00c3%u0003%u8b0
0x0190  3025 7535 3331 6225 7535 3366 6625 7530      0%u531b%u53ff%u0
0x01a0  3037 3825 7530 3030 3025 7530 303d 6120      078%u0000%u00=a.
0x01b0  2048 5454 502f 312e 300d 0a43 6f6e 7465  .HTTP/1.0..Conte
0x01c0  6e74 2d74 7970 653a 2074 6578 742f 786d nt-type:.text/xm
0x01d0  6c0a 484f 5354 3a77 7777 2e77 6f72 6d2e  l.HOST:www.worm.
0x01e0  636f 6d0a 2041 6363 6570 743a 202a 2f2a  com..Accept:.*/*
0x01f0  0a43 6f6e 7465 6e74 2d6c 656e 6774 683a  .Content-length:
0x0200  2033 3536 3920 0d0a 0d0a 558b ec81 ec18 .3569.....U.....
0x0210  0200 0053 5657 8dbd e8fd ffff b986 0000  ...SVW..........
0x0220  00b8 cccc cccc f3ab c785 70fe ffff 0000  ..........p.....
0x0230  0000 e90a 0b00 008f 8568 feff ff8d bdf0  .........h......
0x0240  feff ff64 a100 0000 0089 4708 6489 3d00  ...d......G.d.=.
0x0250  0000 00e9 6f0a 0000 8f85 60fe ffff c785  ....o.....`.....
0x0260  f0fe ffff ffff ffff 8b85 68fe ffff 83e8  ..........h.....
0x0270  0789 85f4 feff ffc7 8558 feff ff00 00e0  .........X......
0x0280  77e8 9b0a 0000 83bd 70fe ffff 000f 85dd  w.......p.......
0x0290  0100 008b 8d58 feff ff81 c100 0001 0089  .....X..........
0x02a0  8d58 feff ff81 bd58 feff ff00 0000 7875  .X.....X......xu
0x02b0  0ac7 8558 feff ff00 00f0 bf8b 9558 feff  ...X.........X..
0x02c0  ff33 c066 8b02 3d4d 5a00 000f 859a 0100  .3.f..=MZ.......
0x02d0  008b 8d58 feff ff8b 513c 8b85 58fe ffff  ...X....Q<..X...
0x02e0  33c9 668b 0c10 81f9 5045 0000 0f85 7901 3.f.....PE....y.
0x02f0  0000 8b95 58fe ffff 8b42 3c8b 8d58 feff  ....X....B<..X..
0x0300  ff8b 5401 7803 9558 feff ff89 9554 feff  ..T.x..X.....T..
0x0310  ff8b 8554 feff ff8b 480c 038d 58fe ffff  ...T....H...X...
0x0320  898d 4cfe ffff 8b95 4cfe ffff 813a 4b45  ..L.....L.....:KE
0x0330  524e 0f85 3301 0000 8b85 4cfe ffff 8178  RN..3.....L....x
0x0340  0445 4c33 320f 8520 0100 008b 8d58 feff  .EL32........X..
0x0350  ff89 8d34 feff ff8b 9554 feff ff8b 8558  ...4.....T.....X
0x0360  feff ff03 4220 8985 4cfe ffff c785 48fe  ....B...L.....H.
0x0370  ffff 0000 0000 eb1e 8b8d 48fe ffff 83c1  ..........H.....
0x0380  0189 8d48 feff ff8b 954c feff ff83 c204  ...H.....L......
0x0390  8995 4cfe ffff 8b85 54fe ffff 8b8d 48fe  ..L.....T.....H.
0x03a0  ffff 3b48 180f 8dc0 0000 008b 954c feff  ..;H.........L..
0x03b0  ff8b 028b 8d58 feff ff81 3c01 4765 7450  .....X....<.GetP
0x03c0  0f85 a000 0000 8b95 4cfe ffff 8b02 8b8d  ........L.......
0x03d0  58fe ffff 817c 0104 726f 6341 0f85 8400  X....|..rocA....
0x03e0  0000 8b95 48fe ffff 0395 48fe ffff 0395  ....H.....H.....
0x03f0  58fe ffff 8b85 54fe ffff 8b48 2433 c066  X.....T....H$3.f
0x0400  8b04 0a89 854c feff ff8b 8d54 feff ff8b  .....L.....T....
0x0410  5110 8b85 4cfe ffff 8d4c 10ff 898d 4cfe  Q...L....L....L.
0x0420  ffff 8b95 4cfe ffff 0395 4cfe ffff 0395  ....L.....L.....
0x0430  4cfe ffff 0395 4cfe ffff 0395 58fe ffff  L.....L.....X...
0x0440  8b85 54fe ffff 8b48 1c8b 140a 8995 4cfe  ..T....H......L.
0x0450  ffff 8b85 4cfe ffff 0385 58fe ffff 8985  ....L.....X.....
0x0460  70fe ffff eb05 e90d ffff ffe9 16fe ffff  p...............
```

```
0x0470   8dbd f0fe ffff 8b47 0864 a300 0000 0083   .......G.d......
0x0480   bd70 feff ff00 7505 e938 0800 00c7 854c   .p....u..8.....L
0x0490   feff ff01 0000 00eb 0f8b 8d4c feff ff83   ...........L....
0x04a0   c101 898d 4cfe ffff 8b95 68fe ffff 0fbe   ....L.....h.....
0x04b0   0285 c00f 848d 0000 008b 8d68 feff ff0f   ...........h....
0x04c0   be11 83fa 0975 218b 8568 feff ff83 c001   .....u!..h......
0x04d0   8bf4 50ff 9590 feff ff3b f490 434b 434b   ..P......;..CKCK
0x04e0   8985 34fe ffff eb2a 8bf4 8b8d 68fe ffff   ..4....*....h...
0x04f0   518b 9534 feff ff52 ff95 70fe ffff 3bf4   Q..4...R..p...;.
0x0500   9043 4b43 4b8b 8d4c feff ff89 848d 8cfe   .CKCK..L........
0x0510   ffff eb0f 8b95 68fe ffff 83c2 0189 9568   ......h........h
0x0520   feff ff8b 8568 feff ff0f be08 85c9 7402   .....h........t.
0x0530   ebe2 8b95 68fe ffff 83c2 0189 9568 feff   ....h........h..
0x0540   ffe9 53ff ffff 8b85 68fe ffff 83c0 0189   ..S.....h.......
0x0550   8568 feff ff8b 4d08 8b91 8400 0000 8995   .h....M.........
0x0560   6cfe ffff c785 4cfe ffff 0400 0000 c685   l.....L.........
0x0570   d0fe ffff 688b 4508 8985 d1fe ffff c785   ....h.E.........
0x0580   d5fe ffff 5b53 53ff c785 d9fe ffff 6378   ....[SS.......cx
0x0590   9090 8b4d 088b 5110 8995 50fe ffff 83bd   ...M..Q...P.....
0x05a0   50fe ffff 0075 268b f46a 008d 854c feff   P....u&..j...L..
0x05b0   ff50 8b8d 68fe ffff 518b 5508             .P..h...Q.U.
```

Description of the Attack

All leads point toward this being CodeRed or a flavor of CodeRed. After examining the packet above and comparing it to the one on cert.org, they are virtually identical. According to cert.org, this capture does not mean that the target has been infected, but that it is attempting to infect it. The only anomaly that I can see is both the MF and DF bits are set. This leads me to believe that it is a specially crafted CodeRed packet. Had this been an actual CodeRed infected target, I would think it would be more active. From the ten days that I imported, these two events are the only ones captured in all the IDS files from this source. Also, had it actually infected the target system, it should have started to make some noise. Just like the source, there were only two events captured in all the IDS files. This does not mean the target didn't get infected. It still could have been and the traffic coming out was just not captured by the IDS. From the data present, it's impossible to tell.

For a more detailed explanation of CodeRed, see the cert.org write-up.

Attack Mechanism

The attack mechanism for CodeRed is it attempts to connect via TCP on port 80 looking for vulnerable IIS web servers. If it finds a server that answers its request, it will send a specially crafted HTTP GET request to the target attempting to exploit a buffer overflow in the indexing service of the IIS web server. If the system becomes infected it will attempt the same process on other randomly selected targets.

Based upon the information present in the IDS file, my guess would be that this is a stimulus packet. Looking through ten days worth of data, there was only two packets sent from this host and both went to the same target. Others could have been sent, but just not captured. Typical behavior of a CodeRed attack would be to attempt infection on multiple random hosts. This does not look like the case here. This would steer me in the direction

of packet crafting. Let's take a closer look at the IP Header. This has been cut from the packet above.

Figure 2.3.2: A closer look at the Headers

```
0x0000  4500 05bc 7d92 6000 6f06 189d d56b 4454          E...}.`.o....kDT
0x0010  cfa6 b0f0 12be 0050 2618 748b 2d96 92a6  .......P&.t.-...
0x0020  8018 faf0 d3a6 0000 0101 080a 0002 9d03  ................
0x0030  55d9 fd67 4745 5420 2f64 6566 6175 6c74  U..gGET./default
```

Let's see if a quick glance at the fields shows anything interesting. The blue represents the IP Header and the red is the TCP Header.

IP Header Information:
45 – This shows a normal IPv4 protocol with a header length of 20 bytes.
00 – We see that all the TOS bits are set to zero.
05bc – Total length of the datagram is 1468 bytes.
7d92 – Fragment ID of 32146
60 – This is the sixth offset which shows the fragmentation bits that are set. When you convert this to binary, you get 0110 which says that the DF and MF bits are set.
00 – Shows that the rest of the Fragment Offset is not set.
6f – This is the TTL value which is set to 111
06 – Protocol is set to TCP
189d – Header Checksum. This has been purposely munged.
Df6b 4454 – Source IP = 213.107.68.84
Cfa6 b0f0 – Destination IP = 207.166.176.240

TCP Header Information:
12be – Source Port = 4798
0050 – Destination Port = 80
2618 748b = Sequence Number = 639137263
2d96 92a6 = Acknowledgement Number = 764842662
80 – Header Length = 8. Multiply this by the protocol in the IP Header to get the total length of the TCP Header. 8 * 4 = 32 bytes long.
18 – These are the flags that are set. To do this part correctly you need to split each one up into 4 bits and convert it into binary. Doing this would result in the 1 = 0001, which would have the ACK flag set and the 8 = 1000, which has the PSH flag set.
Fafo – Window Size = 64240
D3a6 – TCP Checksum. This has also been munged.
0000 – No Urgent Pointer are set.
0101 080a – Options

We can see from the breakdown of the headers that the above assumptions still hold true. The main factor here is both DF and MF bits are set and you cannot do this. Another thing to note here is the header also shows the PSH and ACK flags. In order for a source to push data to a target the three-way handshake needs to be completed. This would further the argument that the source address was not spoofed, as stated above. In my

opinion, this is definitely a CodeRed attack attempting to evade an IDS system or a firewall.

Correlations

Reading through the mailing list, this particular event has been analyzed many times. Below are the links to some of the analysis:

1. http://cert.uni-stuttgart.de/archive/intrusions/2002/08/msg00106.html
2. http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00121.html
3. http://cert.uni-stuttgart.de/archive/intrusions/2002/08/msg00237.html
4. http://cert.uni-stuttgart.de/archive/intrusions/2002/10/msg00218.html

In addition to the mailing list posting above, below are some other links to write-ups and advisories.

1. http://www.cert.org/advisories/CA-2001-19.html
2. http://www.securityfocus.com/bid/2880
3. http://www.microsoft.com/technet/security/bulletin/MS01-044.asp

I believe the above links strengthens my argument that this is an attack designed to evade an IDS or firewall. I also did a Google search for "Fragmented CodeRed", but I didn't find any other information other than what's already noted above. I also went to DShield and did a search on this IP Address. The only thing that showed up was the Whois information. You can view this information in appendix A below. One last thing I did was another Google search on the IP Address and the only return was a link to some SnortSnarf logs. I attempted to view it, but the link was broken.

Evidence of Active Targeting

I don't believe this attack was the result of active targeting. Of the ten days that I inspected, only one of them had data from this IP. In total, there were two events from this source directed towards the same target. No other data was captured. I believe this was a stealthy CodeRed attack. According to CERT's write-up on CodeRed, it attacks randomly chosen IP Addresses.

Just to figure out if either the source or destination was running any form of web services, I used a tool called Netcraft to query the site. Unfortunately, both sites did not return any results.

Severity

Severity is calculated using the formula below:

(Criticality + Lethality) - (System Countermeasures + Network Countermeasures)

Each element is worth 1 to 5 points, and the arithmetic gives us a range of -8 to +8.

Criticality	At this point it's difficult to tell if target system is even a WEB server. If	**3**

	it's not, the criticality could be even lower. Because I don't know, I'm going to assume it is. The fact that no replies or additional attacks were captured from the target tells me that if it is a web server, it's not vulnerable.	
Lethality	Had this host been a vulnerable WEB server, the attack would have compromised the machine. This is not a good thing.	**5**
System	Since no replies or additional attacks were captured, it can be assumed that the target was not vulnerable to this attack. Either this or it has already been patched. Either way, well be a little conservative.	**3**
Network	There's not too much information gathered from the logs. Obviously, more would have been nice. Besides, why would you allow a packet into your network that has both DF and MF bits set.	**1**

Severity = (3+5) – (3+1) = **4**

This gives me an incident severity of **4**. There is some work that needs to be done to harden this environment more.

Defensive Recommendation

First and foremost, patch your system. There has been a patch out for this vulnerability for well over a year. This should be common sense, but apparently it's not. You would then want to have some form of antivirus running real-time scanning. Setup an inner and outer firewall and have some form of content filtering in place. This will ensure that packets with improper flag combinations will not make it into your network. Another good idea might be to block all ICMP traffic that does not have a source address from inside your network. Last, but not least, setup an IDS system to complement your other defenses. Also, make sure your firewalls are stateful firewalls.

Multiple Choice Test Question

Given the following packet header, which offset value represents the fragment offset and which flags are set?

0x0000 4500 05bc 7d92 6000 6f06 189d d56b 4454
0x0010 cfa6 b0f0 12be 0050 2618 748b 2d96 92a6
0x0020 8018 faf0 d3a6 0000 0101 080a 0002 9d03
0x0030 55d9 fd67 4745 5420 2f64 6566 6175 6c74

a) 18 and PSH ACK
b) 60 and DF and MF
c) 00 and 0x0
d) 06 and TCP
e) None of the above

The answer is (b). The fragment offset is located at offset 6 and 7 of the IP Header which is 6000. When you break this value down into its binary, you get 0110 which equals 6.

Starting from right to left, you don't really care right most 0, so you're left with 011 which are the MF and DF.

References

Cert.org. "CERT® Advisory CA-2001-19 "CodeRed" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL" URL: http://www.cert.org/advisories/CA-2001-19.html (January 17, 2002).

Cormier, André. "LOGS: GIAC GCIA Version 3.3 Practical Detect(s)" URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00121.html (January 2003).

Incidents.org. "GIAC Certification Practical Logs" URL: http://www.incidents.org/logs/Raw/2002.10.10 (May 2002).

Incidents.org. "Intrusions GIAC Mailing List Archives" URL: http://www.incidents.org/archives (May 9, 2003).

Gregory, Scott. "LOGS: GIAC GCIA Version 3.2 Practical Detect(s)" URL: http://cert.uni-stuttgart.de/archive/intrusions/2002/08/msg00106.html (August 2002).

Kan, Bernard. "LOGS: GIAC GCIA Version 3.2 Practical Detect #3" URL: http://cert.uni-stuttgart.de/archive/intrusions/2002/10/msg00218.html (October 2002).

Merchant, Corey. "GIAC GCIA – Fragmented CodeRed" URL: http://cert.uni-stuttgart.de/archive/intrusions/2002/08/msg00237.html (August 2002).

Microsoft Corporation. **"**Microsoft Security Bulletin MS01-044" URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-044.asp (August 2001).

Netcraft. URL: http://www.netcraft.com/ (May 2003).

Postel, Jon. "RFC791 – Internet Protocol Specification" URL: ftp://ftp.rfc-editor.org/in-notes/rfc791.txt (September 1981).

SecurityFocus. "MS Index Server and Indexing Service ISAPI Extension Buffer Overflow Vulnerability" URL: http://www.securityfocus.com/bid/2880 (August 10, 2001).

Stevens, W. Richard. "TCP/IP Illustrated, Volume 1" Reading: Addison Wesley, Inc, 1994.

Appendix A

DShield Results for Source IP

IP Address: 213.107.68.84
HostName: pc3-oxfd1-3-cust84.oxfd.cable.ntl.com
DShield Profile: Country:

Contact E-mail:

Total Records against IP:

Number of targets:

Date Range:
to

Update Summary
Ports Attacked (up to 10):

Port	Attacks	Start	End

Fightback: not sent
Whois:

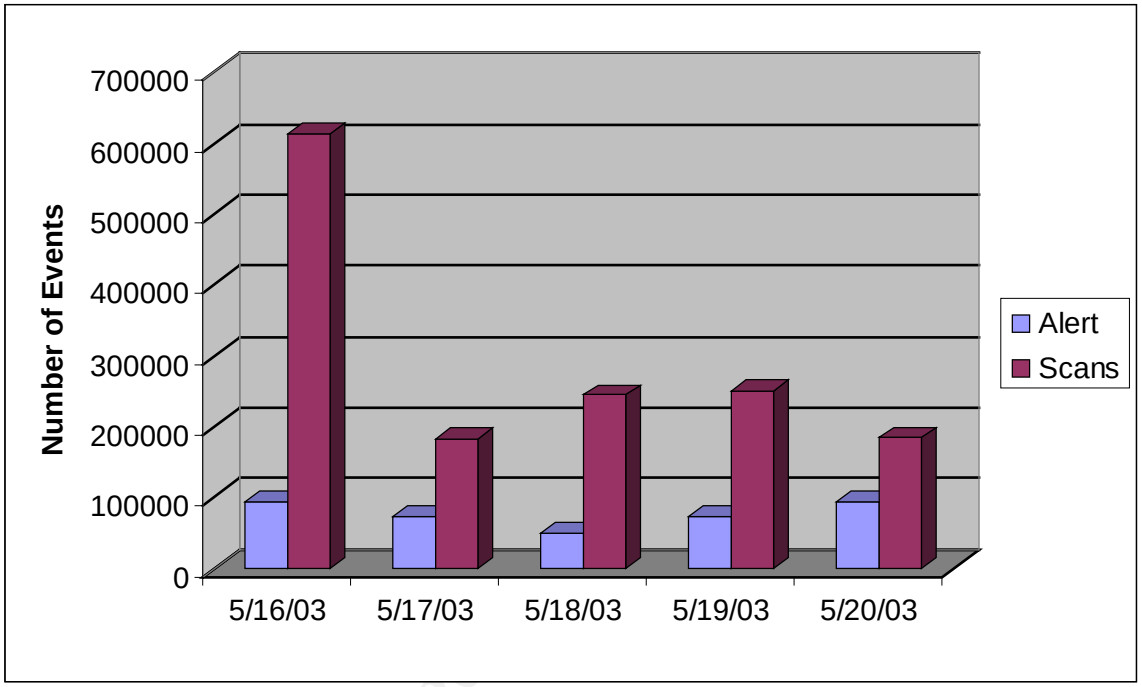
```
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-
services/db/copyright.html

inetnum:      213.107.64.0 - 213.107.71.255
netname:      NTL
descr:        NTL Oxford - CABLE HEADEND
country:      GB
admin-c:      NNMC1-RIPE
tech-c:       NNMC1-RIPE
status:       ASSIGNED PA
mnt-by:       AS5089-MNT
changed:      hostmaster@ntli.net 20011012
```

```
changed:      hostmaster@ntli.net 20020815
source:       RIPE

route:        213.104.0.0/14
descr:        NTL-UK-IP-BLOCK-4
origin:       AS5089
mnt-by:       AS5089-MNT
changed:      bob.procter@ntli.net 20000404
source:       RIPE

role:         NTLI Network Management Centre
address:      NTL Internet
address:      Crawley Court
address:      Winchester
address:      Hampshire
address:      SO21 2QA
phone:        +44 1483 875105
fax-no:       +44 1483 875150
trouble:      ------------------------------------------
---
trouble:      abuse@ntlworld.com - for abuse
notifications
trouble:      nmc@ntli.net - for technical
issues/questions
trouble:      peering@ntli.net - for peering
issues/requests
trouble:      ------------------------------------------
---
e-mail:       nmc@ntli.net
admin-c:      JW1142-RIPE
admin-c:      EH976-RIPE
admin-c:      JM2907-RIPE
admin-c:      HD98-RIPE
admin-c:      AF3217-RIPE
tech-c:       MC1641-RIPE
tech-c:       JW1142-RIPE
tech-c:       EH976-RIPE
tech-c:       JM2907-RIPE
tech-c:       HD98-RIPE
tech-c:       AF3217-RIPE
tech-c:       MN3609-RIPE
tech-c:       AD7775-RIPE
nic-hdl:      NNMC1-RIPE
mnt-by:       AS5089-MNT
notify:       data.planning@ntl.com
notify:       nmc@ntli.net
changed:      hostmaster@ntli.net 20020619
changed:      hostmaster@ntli.net 20020815
changed:      hostmaster@ntli.net 20020913
changed:      hostmaster@ntli.net 20030328
changed:      hostmaster@ntli.net 20030401
source:       RIPE
```

Assignment #3: Analyze This!

Analysis of the University Network

Executive Summary



The above graph represents the number of events for the five day period of May 16th through the 20th 2003.

From the graph above we can see a fairly consistent flow of events except for Friday the 16th where we see a huge spike in the number of the scans that occurred in the University's network. This spike could be the start of a reconnaissance effort or the end of a major attack suffered by the University.

The goal of this paper is to provide an understanding as to the type of events seen on the University's network. With this information the University can plan additional detection and prevention strategies, as well as pursue tuning recommendations and mitigation tasks.

It is fairly evident that the University has some security safeguards in place. In lieu of that, it appears that either Nimda or Code Red is running rampant within the network. Furthermore, it also appears that other hosts have been compromised by Trojans and backdoors being run by users via IRC Chat channels.

Logs Analyzed

The following is the list of files I chose to analyze for the third assignment. They are for five consecutive days and were retrieved from http://www.incidents.org/logs. [10]

Alert Files	Size (Bytes)
alert.030516.gz	2,029,950
alert.030517.gz	1,567,656
alert.030518.gz	1,280,776
alert.030519.gz	1,648,434
alert.030520.gz	1,799,547

The alert files were all concatenated into one file. This was done for simplicity and to make it easier to spot trends in the data. I also took Tod Beardsley's [1] advice and excluded the alerts generated by Snort's portscan preprocessor. Tod stated that this data was available in the raw scan files. I used SnortSnarf [9], snortalog [8], and snort_stat [7] to assist me with the analysis portion of the alert files. These programs were used to generate nice HTML reports of the data.

Scan Files	Size (Bytes)
scan.030516.gz	4,891,968
scan.030517.gz	1,734,772
scan.030518.gz	2,084,334
scan.030519.gz	2,241,774
scan.030520.gz	1,739,255

As with the alert files above, the scan files were also concatenated into one file. Sawmill [5] was used to help parse out the data and spot trends.

OOS Files	Size (Bytes)
OOS_Report_2003_05_16_6191.txt	1,136,643
OOS_Report_2003_05_17_14869.txt	829,443
OOS_Report_2003_05_18_9515.txt	563,203
OOS_Report_2003_05_19_9542.txt	1,116,163
OOS_Report_2003_05_20_14142.txt	640,003

The OOS files are "Out of Spec" files captured from the above scan and alert files. There data will contain some illegal or out of the ordinary combination of bits set. This data will help you correlate your analyses.

Detects List

Below is a list of detects that were generated using SnortSnarf. They are prioritized by number of occurrences. As per the GCIA Practical and Planning Guide, following this will be a brief description of these events, linking the common ones together to help identify relationships, correlations from other practicals, and recommendations towards

actions that should be taken. Also included will be a link graph and analysis followed by a list of five selected external sources and a top talkers list. The list below shows all events that have generated more than 50 events.

Priority	Signature (click for sig info)	Alerts	Sources	Dests
1	SMB Name Wildcard	195845	27246	41429
2	High port 65535 udp - possible Red Worm - traffic	44834	320	367
3	High port 65535 tcp - possible Red Worm - traffic	29881	461	2164
4	Tiny Fragments – Possible Hostile Activity	20225	17	636
5	spp_http_decode: IIS Unicode attack detected	14812	736	899
6	CS WEBSERVER - external web traffic	13118	5832	3
7	[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	10715	10	22
8	External RPC call	10184	2	9405
9	Incomplete Packet Fragments Discarded	8927	136	109
10	TFTP - Internal TCP connection to external tftp server	7834	48	47
11	Possible trojan server activity	5901	80	4118
12	Null scan!	5034	132	142
13	SUNRPC highport access!	2817	37	26
14	spp_http_decode: CGI Null Byte attack detected	2483	119	135
15	Queso fingerprint	1797	321	129
16	IDS552/web-iis_IIS ISAPI Overflow ida nosize [arachNIDS]	977	554	763
17	EXPLOIT x86 NOOP	795	118	130
18	CS WEBSERVER - external ftp traffic	791	181	1
19	TCP SRC and DST outside network	685	146	58
20	[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.	461	58	74
21	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize [arachNIDS]	434	9	385
22	[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	253	14	12
23	[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	226	7	2
24	SNMP public access	224	21	20
25	IRC evil - running XDCC	169	13	18
26	NIMDA - Attempt to execute cmd from campus host	150	8	147
27	EXPLOIT x86 setuid 0	146	136	117
28	NMAP TCP ping!	145	55	62
29	connect to 515 from outside	132	1	2
30	TFTP - Internal UDP connection to external tftp server	130	43	37
31	EXPLOIT x86 setgid 0	65	65	61
32	EXPLOIT x86 stealth noop	60	18	10

Alert Details

2	**High port 65535 udp - possible Red Worm – traffic**

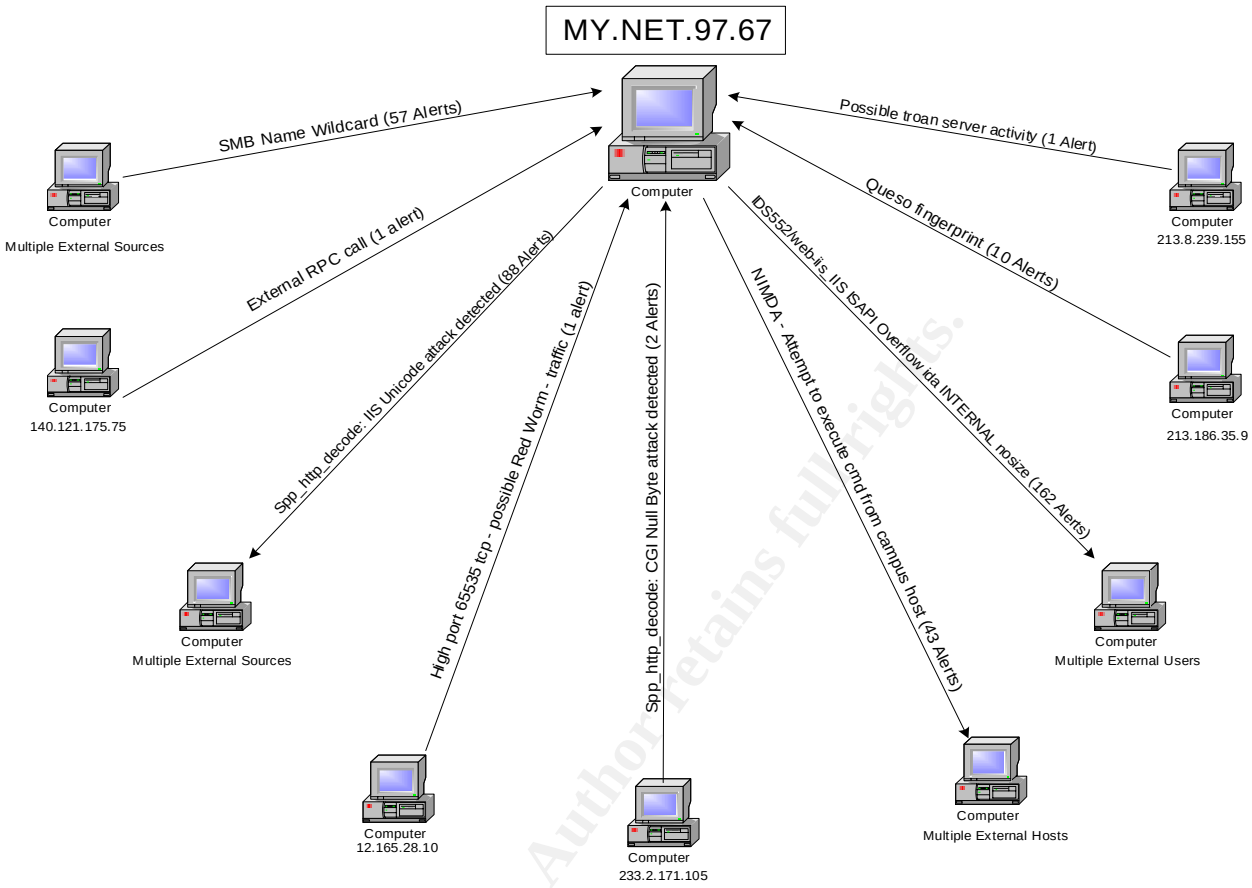
3	**High port 65535 tcp - possible Red Worm – traffic**
5	**spp_http_decode: IIS Unicode attack detected**
16	**IDS552/web-iis_IIS ISAPI Overflow ida nosize [arachNIDS]**
21	**IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize [arachNIDS]**

```
05/16-01:26:58.966524  [**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.201.58:65535 -
> 66.124.36.48:5122
05/16-01:26:58.966537  [**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.201.58:65535 -
> 81.79.27.62:5121

05/16-04:44:10.491778  [**] High port 65535 tcp - possible Red Worm - traffic [**] 66.24.224.113:65535 ->
MY.NET.209.78:1214
05/16-04:44:10.496063  [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.209.78:1214 ->
66.24.224.113:65535

05/16-07:36:08.139678  [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.97.10:3186 ->
211.233.29.2:80
05/16-08:00:05.390397  [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.97.49:3112 ->
207.200.86.66:80

05/18-09:21:58.755909  [**] IDS552/web-iis_IIS ISAPI Overflow ida nosize [**] 218.63.152.220:2038 ->
MY.NET.70.205:80
05/18-09:22:34.883952  [**] IDS552/web-iis_IIS ISAPI Overflow ida nosize [**] 217.219.241.58:1338 ->
MY.NET.251.128:80

05/17-14:33:06.595300  [**] IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize [**]
MY.NET.97.122:2183 -> 130.205.8.217:80
05/17-14:33:25.295491  [**] IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize [**]
MY.NET.97.122:2371 -> 130.94.189.147:80
```

As you can see, there is a lot of worm activity on the Universities network. Unfortunately, most of these events are associated with Code Red, Code Red II or Nimda. Combined they had a total of 90,938 events. According to CERT [3], "The "Code Red" worm is self-replicating malicious code that exploits a known vulnerability in Microsoft IIS servers (CA-2001-13)".[2]

One major characteristic of Code Red and Nimda is that once they are prevalent in your network they are very noisy. You can see from events they generate they are obviously infected. I have attempted to show this in the graphs below.

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
MY.NET.97.67	54	210	54	170
MY.NET.97.19	43	226	42	180
MY.NET.97.44	19	48	19	46
MY.NET.98.146	18	98	18	84
MY.NET.97.18	11	29	10	24
MY.NET.97.122	3	10	2	8
MY.NET.97.165	1	33	1	8
MY.NET.122.35	1	1	1	1



Link Diagram for Possible Nimda Activity

As you can see from the graph above, Code Red and Nimda infected hosts can generate a lot of different alerts. You can start to see a pattern of scans coming in and scans going out. One thing to note here is that some of the systems in the graph did not generate a lot of events. These particular systems don't follow the expected patterns and might be false-positives. Below are the top 5 external Code Red or Nimda sources.

IP Address: 211.233.29.4
HostName: 211.233.29.4
DShield Profile: Country:
KR

Contact E-mail:
ip@kidc.net

Total Records against IP:
not processed

Number of targets:
select update below

Date Range:

to

Update Summary

Top 10 Ports hit by this source:

Port
Attacks
Start
End

Last Fightback Sent: not sent

Whois:

```
IP Address          : 211.233.28.0-211.233.31.255
Connect ISP Name    : KIDC
Connect Date        : 20001213
Registration Date   : 20011115
Network Name        : KIDC-INFRA-SERVERROOM-DAUM

[ Organization Information ]
Orgnization ID      : ORG231919
Name                : Daum Communication
State               : SEOUL
Address             : Gangnam-gu, Yeoksam-dong, DACOM
B/D 12F. 706-1
Zip Code            : 135-987

[ Admin Contact Information]
Name                : Hanju Kim
Org Name            : Daum Communication
State               : SEOUL
Address             : Gangnam-gu, Yeoksam-dong, DACOM
B/D 12F. 706-1
Zip Code            : 135-987
Phone               : +82-2-6446-6407
Fax                 : +82-2-6446-6499
E-Mail              : hankim@daumcorp.com

[ Technical Contact Information ]
Name                : youngchul Lee
Org Name            : Daum Communication
State               : SEOUL
Address             : Gangnam-gu, Yeoksam-dong, DACOM
B/D 12F. 706-1
```

```
Zip Code            : 135-987
Phone               : +82-2-6446-6407
Fax                 : +82-2-6446-6499
E-Mail              : uniace@daumcorp.com
```

IP Address: 218.109.54.61
HostName: 218.109.54.61
DShield Profile: Country:

Contact E-mail:

Total Records against IP:
not processed

Number of targets:
select update below

Date Range:
to

Update Summary

Top 10 Ports hit by this source:

Port	Attacks	Start	End

Last Fightback Sent: not sent

Whois:

```
inetnum:      218.0.0.0 - 218.255.255.255
netname:      APNIC-AP
country:      AU
descr:        Asia Pacific Network Information Center,
Pty. Ltd.
              Regional Internet Registry for the Asia-
Pacific Region
              Level 1 - 33 Park Road.
              PO Box 2131
              Milton QLD 4064
              Australia
```

```
admin_c:      HM20-AP
tech_c:       NO4-AP
remarks:      Unresolved Spam complaints to Auto-
responder spam@apnic.net.
              Unresolved Network Abuse issues to Auto-
responder
              abuse@apnic.net.
mnt_by:       APNIC-HM
changed:      dbmon@apnic.net 20010801
              hm-changed@apnic.net 20021001
status:       ALLOCATED PORTABLE
source:       APNIC
notify:
mnt_lower:    APNIC-HM
rev_srv:
start:        3657433088
end:          3674210303
diff:         16777215
person:       APNIC Network Operations
address:      Level 1
              33 Park Road
              Milton QLD 4064
country:      AU
phone:        +61 7 3858 3100
fax_no:       +61 7 3858 3199
e_mail:       technical@apnic.net
nic_hdl:      NO4-AP
mnt_by:       MAINT-APNIC-AP
changed:      technical@apnic.net 19981111
              hostmaster@apnic.net 20020211
source:       APNIC
remarks:      Administrator for APNIC Network
Operations
notify:       dbmon@apnic.net
```

IP Address: 211.233.29.2
HostName: 211.233.29.2
DShield Profile: Country:
KR

Contact E-mail:
ip@kidc.net

Total Records against IP:
not processed

Number of targets:
select update below

Date Range:

to

Update Summary

Top 10 Ports hit by this source:

Port	Attacks	Start	End

Last Fightback Sent: not sent

Whois:

```
IP Address          : 211.233.28.0-211.233.31.255
Connect ISP Name    : KIDC
Connect Date        : 20001213
Registration Date   : 20011115
Network Name        : KIDC-INFRA-SERVERROOM-DAUM

[ Organization Information ]
Orgnization ID      : ORG231919
Name                : Daum Communication
State               : SEOUL
Address             : Gangnam-gu, Yeoksam-dong, DACOM
B/D 12F. 706-1
Zip Code            : 135-987

[ Admin Contact Information]
Name                : Hanju Kim
Org Name            : Daum Communication
State               : SEOUL
Address             : Gangnam-gu, Yeoksam-dong, DACOM
B/D 12F. 706-1
Zip Code            : 135-987
Phone               : +82-2-6446-6407
Fax                 : +82-2-6446-6499
E-Mail              : hankim@daumcorp.com

[ Technical Contact Information ]
Name                : youngchul Lee
Org Name            : Daum Communication
State               : SEOUL
Address             : Gangnam-gu, Yeoksam-dong, DACOM
B/D 12F. 706-1
Zip Code            : 135-987
Phone               : +82-2-6446-6407
Fax                 : +82-2-6446-6499
E-Mail              : uniace@daumcorp.com
```

IP Address: 216.35.123.105
HostName: boards.theforce.net
DShield Profile: Country:
US

Contact E-mail:
CompServ@Exodus.net

Total Records against IP:
not processed

Number of targets:
select update below

Date Range:
to

Update Summary

Top 10 Ports hit by this source:

Port	Attacks	Start	End

Last Fightback Sent: sent to CompServ@Exodus.net on 2002-05-25 03:23:49
no reply received

Whois:

```
OrgName:    Cable & Wireless
OrgID:      EXCW
Address:    3300 Regency Pkwy
City:       Cary
StateProv:  NC
PostalCode: 27511
Country:    US

NetRange:   216.32.0.0 - 216.35.255.255
CIDR:       216.32.0.0/14
NetName:    LEGACY-8
NetHandle:  NET-216-32-0-0-1
Parent:     NET-216-0-0-0-0
NetType:    Direct Allocation
```

```
NameServer: DNS01.EXODUS.NET
NameServer: DNS02.EXODUS.NET
NameServer: DNS03.EXODUS.NET
NameServer: DNS04.EXODUS.NET
Comment:    * Rwhois reassignment information for this
block is available at:
Comment:    * rwhois.exodus.net 4321
Comment:    * For abuse please contact abuse@exodus.net
RegDate:    1998-07-30
Updated:    2002-10-30

TechHandle: ZC221-ARIN
TechName:   Cable & Wireless
TechPhone:  +1-919-465-4023
TechEmail:  ip@gnoc.cw.net

OrgAbuseHandle: ABUSE11-ARIN
OrgAbuseName:   Abuse
OrgAbusePhone:  +1-877-393-7878
OrgAbuseEmail:  abuse@exodus.net

OrgNOCHandle: NOC99-ARIN
OrgNOCName:   Network Operations Center
OrgNOCPhone:  +1-800-977-4662
OrgNOCEmail:  trouble@cw.net

OrgTechHandle: EIAA-ARIN
OrgTechName:   Exodus IP Address Administration
OrgTechPhone:  +1-888-239-6387
OrgTechEmail:  ipaddressadmin@exodus.net

OrgTechHandle: GIAA-ARIN
OrgTechName:   Global IP Address Administration
OrgTechPhone:  +1-919-465-4096
OrgTechEmail:  ip@gnoc.cw.net

# ARIN WHOIS database, last updated 2003-06-28 21:05
# Enter ? for additional hints on searching ARIN's WHOIS
database.


OrgName:    American Registry for Internet Numbers
OrgID:      ARIN
Address:    3635 Concorde Parkway, Suite 200
City:       Chantilly
StateProv:  VA
PostalCode: 20151
Country:    US

NetRange:   216.0.0.0 - 216.255.255.255
CIDR:       216.0.0.0/8
NetName:    NET216
NetHandle:  NET-216-0-0-0-0
Parent:
NetType:    Allocated to ARIN
NameServer: ARROWROOT.ARIN.NET
NameServer: BUCHU.ARIN.NET
```

```
NameServer: CHIA.ARIN.NET
NameServer: DILL.ARIN.NET
NameServer: EPAZOTE.ARIN.NET
NameServer: FIGWORT.ARIN.NET
NameServer: GINSENG.ARIN.NET
NameServer: HENNA.ARIN.NET
NameServer: INDIGO.ARIN.NET
Comment:
RegDate:    1998-04-01
Updated:    2002-08-23

OrgNOCHandle: ARINN-ARIN
OrgNOCName:   ARIN NOC
OrgNOCPhone:  +1-703-227-9840
OrgNOCEmail:  noc@arin.net

OrgTechHandle: ARIN-HOSTMASTER
OrgTechName:   Registration Services Department
OrgTechPhone:  +1-703-227-0660
OrgTechEmail:  hostmaster@arin.net

# ARIN WHOIS database, last updated 2003-06-28 21:05
# Enter ? for additional hints on searching ARIN's WHOIS
database.


OrgName:    Cable & Wireless
OrgID:      EXCW
Address:    3300 Regency Pkwy
City:       Cary
StateProv:  NC
PostalCode: 27511
Country:    US
Comment:
Comment:    Rwhois reassignment information for this
block is available at:
Comment:    rwhois.exodus.net 4321
RegDate:
Updated:    2002-08-26

AbuseHandle: ABUSE11-ARIN
AbuseName:   Abuse
AbusePhone:  +1-877-393-7878
AbuseEmail:  abuse@exodus.net

AdminHandle: HINMA-ARIN
AdminName:   Hinman, Tanya
AdminPhone:  +1-800-977-4662
AdminEmail:  thinman@cw.net

NOCHandle: NOC99-ARIN
NOCName:   Network Operations Center
NOCPhone:  +1-800-977-4662
NOCEmail:  trouble@cw.net

TechHandle: EIAA-ARIN
TechName:   Exodus IP Address Administration
```

```
TechPhone:  +1-888-239-6387
TechEmail:  ipaddressadmin@exodus.net

TechHandle: GIAA-ARIN
TechName:   Global IP Address Administration
TechPhone:  +1-919-465-4096
TechEmail:  ip@gnoc.cw.net
```

IP Address: 211.233.29.58
HostName: 211.233.29.58
DShield Profile: Country:
KR

Contact E-mail:
ip@kidc.net

Total Records against IP:
not processed

Number of targets:
select update below

Date Range:
to

Update Summary

Top 10 Ports hit by this source:

Port	Attacks	Start	End

Last Fightback Sent: not sent
Whois:

```
IP Address          : 211.233.28.0-211.233.31.255
Connect ISP Name    : KIDC
Connect Date        : 20001213
Registration Date   : 20011115
Network Name        : KIDC-INFRA-SERVERROOM-DAUM
```

```
[ Organization Information ]
Orgnization ID      : ORG231919
Name                : Daum Communication
State               : SEOUL
Address             : Gangnam-gu, Yeoksam-dong, DACOM
B/D 12F. 706-1
Zip Code            : 135-987

[ Admin Contact Information]
Name                : Hanju Kim
Org Name            : Daum Communication
State               : SEOUL
Address             : Gangnam-gu, Yeoksam-dong, DACOM
B/D 12F. 706-1
Zip Code            : 135-987
Phone               : +82-2-6446-6407
Fax                 : +82-2-6446-6499
E-Mail              : hankim@daumcorp.com

[ Technical Contact Information ]
Name                : youngchul Lee
Org Name            : Daum Communication
State               : SEOUL
Address             : Gangnam-gu, Yeoksam-dong, DACOM
B/D 12F. 706-1
Zip Code            : 135-987
Phone               : +82-2-6446-6407
Fax                 : +82-2-6446-6499
E-Mail              : uniace@daumcorp.com
```

Correlation: Joe Ellis [18] notes this event in his analysis. In his analysis, Joe also noted that hosts that generated a low number of alerts were probably false-positives.

Recommendation: Any and all machines that are infected with either Code Red or Nimda need to be removed from the network and cleaned. If possible, these machines should probably be rebuilt as they might have additional compromises installed. In addition to these steps, all present and future system should have the latest Anti-Virus products installed along with the most up-to-date patches and service packs. One final step would be to setup ingress/egress filters on the outer firewall and drop the packets that match this signature.

26	NIMDA - Attempt to execute cmd from campus host

```
05/17-01:08:16.790056  [**] NIMDA - Attempt to execute cmd from campus host [**] MY.NET.97.44:2468 ->
169.132.41.77:80
05/17-14:03:43.522940  [**] NIMDA - Attempt to execute cmd from campus host [**] MY.NET.97.44:2075 ->
130.149.124.100:80
```

Nimda was a nasty worm that was discovered back in September of 2001. It had many components that attributed to its rapid spread. Some of those were: email, network shares, compromised web shares, active scanning and exploiting through directory transversals. It also looked for backdoors left behind from the Code Red II and sadmind/IIS worms.[4]

Looking through the data it is very apparent that the University has many issues that need to be addressed. From what I'm seeing, these worms are running ramped.

Correlation: Tod Beardsley [1] noted this type of event in his analysis of Nimda. Tod recommended that the University should be dropping these packets as part of both ingress and egress filtering.

Recommendation: Tod is exactly right in his recommendation to drop these packets with both ingress and egress filtering. In addition, the University should check the eight sources that generated this event. They were predominantly on the MY.NET.97.x network but a few stragglers managed to get infected on other subnets. All these systems need to be revisited so they can be cleaned and patched. In all likelihood, they should probably be rebuilt from scratch to ensure nothing is missed. In addition to either method chosen, all service packs and patches should be installed.

1	**SMB Name Wildcard**

```
05/16-01:16:24.354654  [**] SMB Name Wildcard [**] 196.15.187.132:1029 -> MY.NET.214.137:137
05/16-00:54:35.854977  [**] SMB Name Wildcard [**] 12.82.68.64:1026 -> MY.NET.195.207:137
```

These events are the results of increase scanning for the NetBIOS SMB service. According to Bryce Alexander [11], there are two sources for this kind of traffic. The first is an increase in awareness among script kiddies with the ability to discover information about a target host using the NBTSTAT utility. The second would be the spread of an internet worm known as network.vbs [12]. Both of these will use the standard NetBIOS "nbstat" frames, which will elicit a node status response from either NetBIOS or SAMBA clients. This response contains a listing of all NetBIOS names known to that node. [12]

Correlation: Chris Grout [13] noted this event in his analysis. He said that by the speed of the scan and the number of targets that were scanned, it was obviously done by a script-kiddy utilizing one of the many tools available. In all there were 27,246 external sources that scanned 41,429 targets in the University.

Recommendation: The most obvious solution is to set your outer firewall to drop all inbound connection attempts to any NetBIOS port. This would include both TCP and UDP and port 135-139.

4	**Tiny Fragments – Possible Hostile Activity**

```
05/16-05:29:07.373988  [**] Tiny Fragments - Possible Hostile Activity [**] MY.NET.235.110 ->
80.126.52.171
05/16-05:29:07.452076  [**] Tiny Fragments - Possible Hostile Activity [**] MY.NET.235.110 ->
80.126.52.171
```

The Tiny Fragments alert is raised by the snort minfrag preprocessor when it detects a fragmented IP packet that has its threshold set lower than the defined value. According to Marty Roesch, [15] the concept of the preprocessor is that there is no commercial network equipment that fragments their traffic less than 256 bytes, so any traffic you see below

this threshold should be treated as "very suspicious". He also goes on to say that nmap and fragrouter can send packets at 8 or 24 byte fragments.

Correlation: Mark Embrich [14] noted this in his analysis. He said that by setting the fragment lower, you could attempt to bypass a firewall and/or perform a DoS on an unsuspecting host.

Recommendation: Looking at the sources and targets that have been generating this alert, there appears to be one main source, MY.NET.235.110, generating the majority of these events. Of the 20,225 alerts, MY.NET.235.110 has generated 19,624 alerts to 618 distinct external IP's. It's hard to say if this system is infected with something or if this is legit traffic. The number of event seems excessive, but I do not know that threshold value either.

For starters, I would definitely have someone check MY.NET.235.110 to make sure it's not infected with any worm/Trojan. Also ensure that it is up-to-date with the latest service packs and patches and that the anti-virus definitions files are current. Another interesting note is that this host is number one on the top ten talkers list for source hosts – see below. Lastly, I would set Ingress and Egress filters to drop fragments that meet this criteria.

6	**CS WEBSERVER – external web traffic**
18	**CS WEBSERVER – external ftp traffic**

```
05/16-05:31:50.987599  [**] CS WEBSERVER - external web traffic [**] 66.196.72.46:15452 -> MY.NET.100.165:80
05/16-04:55:31.500724  [**] CS WEBSERVER - external web traffic [**] 24.24.244.196:1444 -> MY.NET.100.165:80

05/16-04:42:14.143145  [**] CS WEBSERVER - external ftp traffic [**] 62.101.126.49:3758 -> MY.NET.100.165:21
05/16-04:42:14.143145  [**] CS WEBSERVER - external ftp traffic [**] 62.101.126.49:3758 -> MY.NET.100.165:21
```

This event signifies external web or ftp traffic coming into your internal network. There were 13,118 web alerts generated by 5,832 external sources to mainly one target system – MY.NET.100.165. For the ftp traffic, there were 791 alerts from 118 external sources to one internal system – MY.NET.100.165. My guess is that this server is externally facing and the rule is designed to fire when external traffic accesses the server.

Correlation: Mark Embrich [14] also noted this alert in his analysis. He stated that he was unable to ascertain any information on the rule and why it was triggering. He also suggested that it was probably a custom rule designed to fire when external traffic accessed it.

Recommendation: Since it is not really known why this rule is in existence, University staff will need to examine the server and verify that it is current in patching and virus definitions. Then I would attempt to track down the rule and figure out why it is firing. My guess is it's probably a custom rule designed to log all external traffic that accesses it. It might have been created by the Computer Science department. Maybe that's what the CS stands for. The University staff should check with the Computer Science department

and verify that whether this traffic is expected and within compliance or if it's out of speck. If it's normal and expected, it might be ok to go ahead and remove or modify the rule to help reduce the number of events generated by the IDS.

7	**[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC**
20	**[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan**
22	**[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected**
23	**[UMBC NIDS IRC Alert] Possible sdbot floodnet attempting to IRC**
25	**IRC evil – running XDCC**

05/16-11:04:44.107458 [**] [UMBC NIDS IRC Alert] XDCC client detected attempting to IRC [**]
MY.NET.198.221:1843 -> 205.188.149.12:6667
05/16-11:16:18.165709 [**] [UMBC NIDS IRC Alert] XDCC client detected attempting to IRC [**]
MY.NET.198.221:2576 -> 205.188.149.12:6667

05/16-17:14:10.684064 [**] [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan. [**]
216.152.64.155:6663 -> MY.NET.97.79:1071
05/16-17:40:43.083582 [**] [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan. [**]
216.152.64.155:6662 -> MY.NET.97.79:1592

05/16-12:06:18.231852 [**] [UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected. [**]
206.84.2.2:6667 -> MY.NET.105.204:1673
05/16-12:13:33.950503 [**] [UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected. [**]
206.84.2.2:6667 -> MY.NET.105.204:1673

05/16-21:53:16.474158 [**] [UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC [**]
MY.NET.97.93:3309 -> 216.152.64.155:6666
05/16-22:17:10.955404 [**] [UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC [**]
MY.NET.97.93:3913 -> 216.152.64.155:6666

05/16-07:36:04.569561 [**] IRC evil - running XDCC [**] MY.NET.207.78:2091 -> 217.17.33.10:6667
05/16-07:28:04.568168 [**] IRC evil - running XDCC [**] MY.NET.207.78:2091 -> 217.17.33.10:6667

XDCC client detected attempting to IRC	IRC user /kill detected, possible trojan	Possible Incoming XDCC Send Request Detected	Possible sdbot floodnet attempting to IRC	IRC evil – running XDCC
MY.NET.198.221	There are over 74 different target systems generating this alert for MY.NET.x.x	MY.NET.105.204	MY.NET.97.79	MY.NET.150.205
MY.NET.83.100		MY.NET.241.246	MY.NET.97.85	MY.NET.207.78
MY.NET.88.163		MY.NET.207.78	MY.NET.97.93	MY.NET.241.246
MY.NET.194.125		MY.NET.201.34	MY.NET.97.97	MY.NET.86.33
MY.NET.83.173		MY.NET.80.209	MY.NET.97.242	MY.NET.80.209
MY.NET.252.82		MY.NET.132.24	MY.NET.97.72	MY.NET.249.250
MY.NET.105.204		MY.NET.217.174	MY.NET.202.26	MY.NET.132.24
MY.NET.80.209		MY.NET.150.205		MY.NET.112.199
MY.NET.132.24		MY.NET.201.158		MY.NET.198.221
		MY.NET.211.14		MY.NET.194.125
		MY.NET.252.134		MY.NET.211.14
		MY.NET.112.199		MY.NET.223.130
				MY.NET.201.158

As you can see there is a lot of IRC Chatting going on. This is bad news because IRC is a forum for sending viruses, worms, trojans, backdoors, you name it.

XDCC bots are common sights in channels these days. An XDCC is a bot that has certain packets uploaded to it. These packets may be anything from the recent game to a good

movie. XDCCs are usually r00ted (hacked), and transfer at very high speeds because they are on fast lines. [17]

What has probably happened here is someone is either running some form of P2P software and they have successfully installed an IRC trojan or the students are using IRC to communicate amongst each other or with the outside. In either case, this has led to a multitude of IRC traffic on the University network.

Correlations: Marcus Wu [16] noted similar events in his analysis. In his analysis he is figuring that one of the University host was probably running Gnutella, download and ran a file containing a trojan, which in-turn compromised the Universities network.

Recommendation: It must be assumed that all hosts generating these events have been compromised. The first thing the University needs to do is disconnect them from the network to stop the potential spread. The next step would be to attempt to block all IRC traffic (ports 6665-6669) at the outer firewall. I would probably also include well known ports for P2P applications. The next step would be to install a fresh OS on each machine making sure you install all the latest service packs and patches. Next, make sure that Anti-Virus software is installed with the latest definition files.

Marcus [16] did make another good suggestion. He said the University should keep all sensitive computers on a trusted, more secure network. This is a very good idea. By separating trusted and un-trusted networks, you can reduce the risk of infection to more critical systems. Also, you can lock them down with an additional firewall and another IDS system.

8	**External RPC call**

```
05/16-06:40:27.393263  [**] External RPC call [**] 140.121.175.75:63567 ->
MY.NET.2.163:111
05/16-06:40:27.395440  [**] External RPC call [**] 140.121.175.75:63569 ->
MY.NET.2.165:111
```

The Remote Procedure Call allows a Unix program to execute a command remotely on a Unix system. By doing this, one is also able to map the services that are available. You can see from the sample above, all the external calls are to the portmapper port 111. This port has been a common source of attacks in the past for discovering offered services. This particular reconnaissance attempt is usually followed up by an Attempted Sun RPC high port access. If you see this alert, the initial target system has responded to the external call and advertised which open ports it has to offer. [20] This is considered bad and should definitely be investigated.

In total, the University had 10,184 alerts of this type by mainly one external source address to 9,404 different target systems over a two hour period. The one external source is from the Ministry of Education Computer Center, National Taiwan Ocean University

```
Ministry of Education Computer Center TANET BNETA (NET-140-117-0-0-1)
                                  140.117.0.0  140.138.255.255

National Taiwan Ocean University TANET BNTOU (NET-140-121-0-0-1)
                                  140.121.0.0  140.121.255.255
```

Correlation: Mark Menke [16] noted this event in his analysis. He concluded that the type of reconnaissance was the first step in gathering information. He went on to say this type of attack is usually followed up by an Attempted Sun RPC high port access alert.

Recommendation: More investigation needs to be done. The first step would be to search for any responses to the source as these servers might have vulnerabilities associated with this. The next step would be to compile a list of these servers and have the Universities security team check these servers for out-dated patches and anti-virus software. Lastly, and probably the first step, the University should block on the outer firewalls all external traffic directed towards port 111.

9	**Incomplete Packet Fragments Discarded**

05/18-01:23:31.331056 [**] Incomplete Packet Fragments Discarded [**]
64.152.108.145:0 -> MY.NET.218.130:0
05/18-01:23:31.738689 [**] Incomplete Packet Fragments Discarded [**]
64.152.108.145:0 -> MY.NET.218.130:0

Traffic of this nature tends to be suspicious and should be treated that way. Sending incomplete packets is one way an attacker will attempt to circumvent a firewall or IDS system. One thing to look closely at here is the source and target ports. If they are both 0, you can almost bet there's some crafting going on.

These events are logged by the Snort defragmentation preprocessor, which triggers when received fragments from an 8k or larger packet do no not sum more than half the packet when the last fragment is received. Detects can indicate transmission errors, poor routing, broken stacks, or fragmentation attacks. [19]

Correlation: Dan Hawrylkiw [19] noted this is in his analysis. He stated that same sources that were generating Large UDP Traffic were also responsible for the Incomplete Packet Fragments. Dan believed that the source and destination ports of 0 were caused by the preprocessor logging, which will not report on the source or destination ports and he went on to say that your Snort sensors probably did not capture all the fragments thereby causing the alarms.

Recommendation: It is possible that the University might not have a current build of their IDS. They should ensure that they have the latest stable build of Snort with the current rules and preprocessors. Also, the University staff should investigate this traffic and verify that its legitimacy.

29	**Connect to 515 from outside**

```
05/18-20:57:46.097920  [**] connect to 515 from outside [**] 68.54.94.58:677 ->
MY.NET.24.15:515
05/18-20:57:46.432093  [**] connect to 515 from outside [**] 68.54.94.58:677 ->
MY.NET.24.15:515
```

This event generated 131 attempts from an external source to one target on the University network. This event is associated with external hosts connecting to internal hosts on target port 515. This port has some significance as it is the Unix line printer service. This port has had numerous vulnerabilities posted against it. The latest from CVE is CAN-2001-0906. [21] The Ramen worm is has also been known to exploit this vulnerability.

Correlation: Tod Beardsley [1] noted this in his analysis. He stated that unless access to the lpr servers are intended to be tightly controlled, this alert is all but useless. He went on to say that the rule should be deactivated, and access to the lpr servers should be monitored through local syslogs. [1]

Recommendation: Tod is exactly correct in his analysis. To take his recommendation even further, there is no need for this port to be accessible from the outside. The University staff should also block inbound attempts to port 515 from the outer firewall. In addition to this, MY.NET.24.15 should also be investigated to make sure it is up-to-date with the latest security patches and anti-virus and also check this system to verify that it's within compliance and has not been compromised.

10	**TFTP – Internal TCP connection to external tftp server**
30	**TFTP – Internal UDP connection to external tftp server**

```
05/16-00:37:14.880800  [**] TFTP - Internal TCP connection to external tftp serv
er [**] MY.NET.196.161:3246 -> 209.126.214.14:69
05/16-00:47:37.848070  [**] TFTP - Internal TCP connection to external tftp serv
er [**] MY.NET.224.242:1108 -> 64.12.200.163:69

05/16-05:34:37.945897  [**] TFTP - Internal UDP connection to external tftp serv
er [**] MY.NET.206.130:6257 -> 217.125.139.175:69
05/16-05:24:54.520749  [**] TFTP - Internal UDP connection to external tftp serv
er [**] MY.NET.209.206:6257 -> 217.125.139.175:69
```

TFTP is normally used by thin clients, diskless workstations or remote device upgrades such as routers. To see this traffic travel outside the local network is definitely not normal. A quick glance at the target port shows port 69, which is normally used in TFTP. One thing to remember here is that Nimda also used TFTP to transfer its files. Looking through the alerts I found numerous connections between MY.NET and other external addresses. Either these systems are grossly mis-configured or they've been compromised.

Correlation: Joe Ellis [18] noted these events in his analysis. He said that for traffic to travel from an internal network to an external network using TFTP, either there had to be mis-configured router or the system was probably compromised.

Recommendation: All systems exhibiting this behavior either need to checked for a mis-configuration issue or a compromise. In addition to this, the University should setup an ingress/egress filtering at the perimeter firewall and/or router to stop TCP or UDP 69 traffic from entering or exiting the network.

11	**Possible trojan server activity**

```
05/16-01:47:51.991557  [**] Possible trojan server activity [**] 216.226.129.209
:2887 -> MY.NET.237.250:27374
05/16-01:41:55.718934  [**] Possible trojan server activity [**] 64.146.21.142:2
7374 -> MY.NET.249.122:6346
05/16-06:29:36.464208  [**] Possible trojan server activity [**] 64.68.84.143:27
374 -> MY.NET.162.67:80
```

All alerts associated with this event are showing a source or decimation port of 27374. As all of us know, this port is usually associated with either the SubSeven trojan or the Ramen worm. Some analysts might consider all this data malicious. Take a look at the examples above. These three were extracted from the alert file. The first one shows typical behavior of either SubSeven or Ramen. The second one could be SubSeven or Ramen, but more then likely it's some form of P2P file-sharing like gnutella. The last one looks like normal web server traffic.

Correlation: Tod Beardsley [1] noted this type of behavior in his analysis. In his analysis, he saw the same types of distribution between what could be normal web traffic and what could be either SubSeven or Ramen. Tod's recommendation was to verify that all affected system's had the latest anti-virus software installed with the most current signatures files. SubSeven and Ramen are fairly old. There's no reason one should get infected by it.

Recommendation: My recommendation at this time is to investigate all sources that are exhibiting this behavior and clean and patch the system. Since the majority of these servers are critical web servers, some form of HIDS (Host Intrusion Detection System) should be installed.

12	**Null scan!**

```
05/16-01:45:08.992734  [**] Null scan! [**] 217.235.171.204:0 -> MY.NET.97.91:0
05/16-01:45:49.573982  [**] Null scan! [**] 217.235.171.204:0 -> MY.NET.97.91:0
```

The signature for a null scan fires when a packet has no flags set. Another point of interest is the source and destination port of 0. This should definitely indicate abnormal behavior. This event alone is not malicious in nature. It is mainly used for reconnaissance and probably to help identify the operating system. The source and destination ports of 0 are intended to penetrate firewalls. There are numerous tools available that can do this kind of reconnaissance. Probably the most well known is Nmap.

Correlation: Brian Credeur [23] noted this event in his analysis. He said that these packets are intended to penetrate firewalls scrolling for known or listening ports such as trojans or P2P.

Recommendation: These events, as they come, are not malicious and probably don't need immediate attention. Focus should be put towards the more critical alerts as described above. The main defense against these types of packets is to set your firewall to drop them when they have a pattern match.

13	**SUNRPC highport access!**

```
05/16-08:29:54.999661  [**] SUNRPC highport access! [**] 24.125.66.19:6348 -> MY
.NET.252.78:32771
05/16-08:29:55.652802  [**] SUNRPC highport access! [**] 24.125.66.19:6348 -> MY
.NET.252.78:32771
05/16-08:29:55.712721  [**] SUNRPC highport access! [**] 24.125.66.19:6348 -> MY
.NET.252.78:32771
05/16-08:29:55.728224  [**] SUNRPC highport access! [**] 24.125.66.19:6348 -> MY
.NET.252.78:32771
```

This alert is triggered when any attempt is made to connect to UDP port 32770 and above. Solaris rpcbind listens on UDP ports above 32770. This is in addition to the standard rpcbind port of UDP 111 which is usually filtered from external addresses. UDP ports above 32770 might have a better chance of being open.

This alert is also prone to false-positives. Approximately 95% of the alerts are between 24.125.66.19 and MY.NET.252.78. As shown is the sample above, all alerts are from source port 6348, which is a known P2P. You can see the likes of programs such as Limewire and Bearshare that are known to use this port.

Correlation: Mark Menke [20] noted this event in his analysis. Mark also came to the same conclusion that these alerts are prone to false-positives. In Mark's analysis, he concluded that the source of the UDP port probes was coming from Instant Messaging traffic. Reading Mark's write-up I noticed a source port of 4000 which is known for ICQ servers.

Recommendation: The alert alone is not dangerous, but University staff should investigate MY.NET.252.78 for signs of P2P software and remove them. P2P software is a known medium for spreading virus and trojans. In addition, anti-virus software should be installed on the client in case P2P software is re-installed as this will assist in the prevention of infections. One last step the University should take is to document and publish a policy on using P2P applications. This policy should state that P2P software is forbidden and installations of it will not be tolerated.

14	**spp_http_decode: CGI Null Byte attack detected**

```
05/16-00:55:39.828746  [**] spp_http_decode: CGI Null Byte attack detected [**]
MY.NET.234.210:49564 -> 66.135.192.226:80
```

```
05/16-01:07:54.352073  [**] spp_http_decode: CGI Null Byte attack detected [**]
MY.NET.233.142:49609 -> 208.237.178.27:80
```

Basically, if the http decoding routine finds a %00 in an http request, it will alert with this message. Sometimes you may see false positives with sites that use cookies with urlencoded binary data, or if you're scanning port 443 and picking up SSL encrypted traffic. Having the packet dumps is the only way to tell for sure if you have a real attack [24]

Correlation: Joe Ellis [18] noted this event in his analysis. He stated that the %00 can be valid traffic on a web site if the web site is using CGI. Joe also noted that this feature could be turned off by adding the "-cginull" option to the line "preprocessor http_decode:" in Snort's alert.ids file.

Recommendation: These alerts are more then likely false-positives, but there is not conclusive evidence to show this. It is recommended that the University staff evaluate their WEB servers to verify the legitimacy of these events. Also, University staff should evaluate the rule that generated this event as it may need to be refined to reduce false-positives.

15	Queso fingerprint

```
05/16-01:00:50.557074  [**] Queso fingerprint [**] 217.231.154.132:44166 -> MY.N
ET.24.44:80
05/16-04:23:17.436036  [**] Queso fingerprint [**] 138.23.88.132:56398 -> MY.NET
.60.14:80
05/16-03:40:33.402906  [**] Queso fingerprint [**] 81.57.90.18:34741 -> MY.NET.2
18.154:6346
05/16-04:32:02.404463  [**] Queso fingerprint [**] 216.95.201.31:53468 -> MY.NET
.24.22:25
```

This event indicates that a remote user has used the Queso tool to determine the OS of the server. [25] This tool is used for reconnaissance in an attempt to map out a network for a future attack. From the information gained, an attacker can decide if it's worth targeting or not. You can see from the top two events above, these attackers are targeting WEB servers.

Correlation: Michael Holstein [26] noted this type of activity in his analysis. Michael noted that after investigation of the external hosts, he saw a lot of P2P traffic. He concluded that this type of traffic might be normal P2P traffic, but without being able to examine the signature it would be impossible to tell.

Recommendation: I tend to agree with Michael here. After analyzing the sources, I also discovered potential P2P traffic as indicated by the third event. In addition to the potential P2P traffic, I also noticed SMTP traffic from port 25. It is recommended that the servers in question here need to be hardened with all the latest patches and anti-virus software. This will ensure that they are all protected from the latest vulnerabilities.

17	EXPLOIT x86 NOOP
32	EXPLOIT x86 stealth noop

```
05/16-02:36:36.538164  [**] EXPLOIT x86 NOOP [**] 206.24.190.30:80 ->
MY.NET.240
.90:1245
05/16-03:49:07.390587  [**] EXPLOIT x86 NOOP [**] 205.188.68.196:5190 ->
MY.NET.
222.206:1525
05/16-05:54:48.426033  [**] EXPLOIT x86 stealth noop [**] 131.118.254.130:2698 -
> MY.NET.24.8:119
05/16-06:31:54.153066  [**] EXPLOIT x86 stealth noop [**] 131.118.254.130:2691 -
> MY.NET.24.8:119
```

The x86 NOOP is part of an attack on a remote service, an attacker may attempt to take advantage of insecure coding practices in hopes of executing arbitrary code. This is done by using the NOP instructions. The NOP allows an attacker to fill an address space with a large number of NOPs followed by their code of choice. This will allow the "sledding" into the attacker's shellcode. Unfortunately, the x86 NOP can frequently be found in day-to-day traffic, particularly when transferring large files. [27]

The x86 stealth noop rule triggers when a binary pattern appears in the packet contents which match one form of filler-bytes used in buffer overflow attacks. Buffer overflows allow execution of arbitrary code with the privilege level of the affected server process. Unfortunately, this byte pattern can occur naturally in almost any binary data, so file downloads, streaming media, etc can trigger this event. If this traffic appears to be coming from a web or ftp server outside of your network to one of your client machines, it is likely a false alert caused by someone downloading a binary file. If this was directed at a port on one of your machines which is running a server process, you may want to check to see if it has been exploited. [28]

Correlation: Michael Holstein [26] noted the event in his analysis. Michael noted that web active content, such as Macromedia may trigger this event.

Recommendation: Without investigating the payload, it is impossible to tell if this is a true false-positive or not. The University will need to evaluate the payload to determine if this is a true attack or just a false-positive. The University should also check the target hosts and verify that they are up-to-date with the latest patching and anti-virus products.

19	TCP SRC and DST outside network

```
05/20-21:18:59.116507  [**] TCP SRC and DST outside network [**] 24.52.59.50:29728
-> 67.80.77.94:6112
05/20-21:08:01.205412  [**] TCP SRC and DST outside network [**] 68.9.100.219:4845
-> 67.80.77.94:6112
```

What we are seeing here are both source and destination addresses are not part of the MY.NET network. There are a number of things that can potentially cause this. The first one might be misconfigured networking equipment. Another possible cause might be a

network scan with a spoofed source address. Lastly, this could be as simple as the source or target address being different than the $HOME_NET variable on the IDS sensor. Looking at the events above my guess is that this is either a spoofed source address or a misconfigured router. One interesting note about the data above is the target port of 6112. This port is used a lot for online gaming for games such as Diablo and Starcraft. Searching though the other events I also noticed that there were a lot of addresses where the source or target was a 192.168.x.x address. This particular alert is most likely due to $HOME_ NET variable not being defined to include the private address space.

Correlation: Heather Larrieu [30] noted the event in her analysis. She also stated that this event was either a sign of a bad router configuration, or spoofed traffic originating on the home network.

Recommendation: Depending on the exact cause, numerous steps can be taken. If the issue has to do with the private address space, the University staff will need to adjust the $HOME_NET variable in the IDS configuration file. If the issue has to do with a misconfigured router, the University staff will need to reconfigure the router so it functions correctly. Finally, if the issue has to do with a spoofed source address, then the University staff should setup Egress filtering on their outer firewalls/routers to block all non MY.NET traffic from exiting the network.

24	SNMP public access

```
05/16-03:04:37.280557  [**] SNMP public access [**] 147.46.56.20:1025 -> MY.NET.
154.26:161
05/16-03:46:10.189551  [**] SNMP public access [**] 147.46.56.20:1025 -> MY.NET.
154.26:161
```

Simple Network Management Protocol is the protocol used mostly in network management devices. They employ what is called “community strings”. Community strings are what is used to interlink multiple devices together so they can communicate together. Out of the box, these devices are shipped with a default community sting of “public” and “private”. The first is for reading and the second is for writing. SNMP is also a good recon tool for would-be attacks.

This alert indicates that some one is attempting to access these target systems using the community string of public. It is unclear from the data present whether these target systems have SNMP installed or if these are just random attempts. My guess from the low number of alerts (224) is these systems probably have it enabled and were already actively targeted.

This rule is set to fire when a source attempts to access a target system on port 161 pushing the word “public” in the payload.

Correlation: Tod Beardsley [1] noted this traffic in his analysis. He mentioned that SNMP is used by network management devices and they are shipped with default community strings of public and private.

Recommendation: It is good practice to change your community strings to something other then the default. This will ensure that they won't become easy prey for some would-be attacker. If possible, you should instill the use of strong community strings as this will make it harder for attackers to brute force them. Additionally, you should set a policy to change them when you change your other passwords as SNMP is sent over the network in clear text and can easily be sniffed off your wire.

27	**EXPLOIT x86 setuid 0**
31	**EXPLOIT X86 setgid 0**

05/18-18:18:36.836774 [**] EXPLOIT x86 setuid 0 [**] 204.210.160.37:3878 -> MY.NET.88.214:1214
05/18-18:34:29.445743 [**] EXPLOIT x86 setuid 0 [**] 65.96.86.19:4350 -> MY.NET.205.146:2046

05/19-02:01:10.082475 [**] EXPLOIT x86 setgid 0 [**] 67.160.75.110:1542 -> MY.NET.207.254:4141
05/19-02:20:41.671578 [**] EXPLOIT x86 setgid 0 [**] 212.120.105.148:34192 -> MY.NET.202.114:1214

This alert indicates that shellcode was detected that attempts to set the user identity or group identity to 0 (root). If this code is executed successfully, it is possible for the current process to inherit root privileges. However, setuid(2) requires root privileges to be executed in the first place if the current uid is attempting to get a higher privilege level. [31]

The false-positive rates of these events are fairly high. Large binary transfers, certain web traffic, and even mail traffic can trigger this rule, but are not necessarily indicative of actual setuid or setgid code. [31]

Correlation: Michael Holstein [26] noted the event in his analysis. Michael noted that because the pattern matching is so small, it is frequently triggered by a wide variety of non-suspect activity such as a mix of web and KaZaA traffic in addition to other unknowns.

Recommendation: Due to the high false-positive rate of these events, the University staff will need to analyze the payloads of these events to determine if they are real or not. Unfortunately, with the information I am given, I am unable to determine if they are false or not. The University should also check the target systems and verify they are not compromised. While they are there, they should also verify that the systems are patched and up-to-date of all security and anti-virus signatures.

28	**NMAP TCP ping!**

05/17-23:22:40.703797 [**] NMAP TCP ping! [**] 63.211.17.228:80 -> MY.NET.1.3:53

```
05/17-23:22:40.703808  [**] NMAP TCP ping! [**] 63.211.17.228:53 ->
MY.NET.1.3:53
```

An NMAP TCP ping alert is generated when an incoming packet has the TCP ACK flag set and the ACK field set to 0. [32] This is done to stimulate a response from the target to determine if it's live on the network. Looking through the alerts I noticed there to be a lot of HTTP and DNS traffic signaling this might be legitimate traffic.

Correlation: Tod Beardsley [1] noted this traffic in his analysis. He said that in addition to the false-positives, KaZaA filesharing networks also exhibit this type of behavior.

Recommendation: Due to the high probability of false-positives from the alert, it is recommended that this rule be removed. Tod also noted this same recommendation as Nmap versions greater than 2.5BETA do not exhibit this behavior.

Top Talkers List

Below are the Top Ten Source and Destination Hosts based on the number of alerts generated from the **Alert log**. SnortSnarf was used for this analysis.

Source Hosts

Rank	Total # Alerts	**Source IP**	# Signatures triggered	Destinations involved
rank #1	19630 alerts	**MY.NET.235.110**	7 signatures	(618 destination IPs)
rank #2	18989 alerts	**MY.NET.201.58**	2 signatures	(47 destination IPs)
rank #3	10176 alerts	**140.121.175.75**	1 signatures	(9404 destination IPs)
rank #4	9255 alerts	**MY.NET.198.221**	3 signatures	233.2.171.1, 205.188.149.12
rank #5	7824 alerts	**140.142.19.69**	4 signatures	(3 destination IPs)
rank #6	6715 alerts	**12.235.36.52**	4 signatures	(5 destination IPs)
rank #7	5463 alerts	**MY.NET.251.10**	1 signatures	(18 destination IPs)
rank #8	4022 alerts	**MY.NET.202.206**	3 signatures	(4 destination IPs)
rank #9	3755 alerts	**172.17.3.17**	1 signatures	(3520 destination IPs)
rank #10	3613 alerts	**208.45.250.203**	2 signatures	(3421 destination IPs)

Destination Hosts

Rank	Total # Alerts	**Destination IP**	# Signatures triggered	Originating sources
rank #1	14057 alerts	**MY.NET.100.165**	9 signatures	(6073 source IPs)

rank #2	12733 alerts	**MY.NET.201.58**	4 signatures	(29 source IPs)
rank #3	9254 alerts	**205.188.149.12**	3 signatures	MY.NET.198.221
rank #4	7964 alerts	**MY.NET.153.157**	4 signatures	(5 source IPs)
rank #5	7356 alerts	**MY.NET.202.206**	8 signatures	(14 source IPs)
rank #6	4959 alerts	**66.67.29.21**	1 signatures	MY.NET.251.10
rank #7	4790 alerts	**MY.NET.30.4**	3 signatures	(300 source IPs)
rank #8	4060 alerts	**24.157.3.20**	2 signatures	199.35.158.1, MY.NET.201.58
rank #9	3897 alerts	**MY.NET.251.10**	1 signatures	(20 source IPs)
rank #10	3712 alerts	**209.99.32.118**	1 signatures	MY.NET.251.10, MY.NET.201.58

Below are the Top Ten Source and Destination Hosts based on the number of alerts generated from the **Scan log**. Sawmill was used for this analysis.

Rank	Source IP	Total # Source Hits	Target IP	Total # Target Hits
rank #1	**MY.NET.196.193**	477,761 alerts	**MY.NET.196.26**	116,577 alerts
rank #2	**MY.NET.87.50**	77,346 alerts	**12.209.4.105**	3,511 alerts
rank #3	**MY.NET.197.30**	44,689 alerts	**24.42.0.66**	3,149 alerts
rank #4	**MY.NET.225.154**	19,888 alerts	**24.58.88.4**	2,526 alerts
rank #5	**MY.NET.97.19**	16,028 alerts	**65.70.30.236**	2,517 alerts
rank #6	**MY.NET.250.162**	15,918 alerts	**68.70.30.236**	2,452 alerts
rank #7	**MY.NET.97.18**	15,808 alerts	**24.203.250.243**	2,442 alerts
rank #8	**MY.NET.217.62**	15,256 alerts	**24.114.110.139**	2,321 alerts
rank #9	**MY.NET.97.99**	12,296 alerts	**131.191.68.192**	2,035 alerts
rank #10	**MY.NET.220.62**	11,645 alerts	**12.241.164.112**	2,030 alerts

Below are the Top Ten Source Ports and Destination Ports based on the number of hits for each port based on the **Scan log**. Sawmill was also used for this analysis.

Rank	Source Port	Total # of Hits	Target Port	Total # of Hits
rank #1	**27022**	77,348 alerts	**17300**	497,124 alerts
rank #2	**7674**	27,044 alerts	**80**	105,817 alerts
rank #3	**22321**	22,963 alerts	**137**	76,594 alerts
rank #4	**1025**	18,540 alerts	**27005**	43,717 alerts
rank #5	**1027**	16,078 alerts	**445**	42,251 alerts
rank #6	**2328**	15,442 alerts	**23000**	35,257 alerts
rank #7	**1026**	13,425 alerts	**7674**	27,047 alerts

rank #8	**1028**	13,082 alerts	**21**	23,514 alerts
rank #9	**1029**	12,888 alerts	**22321**	22,792 alerts
rank #10	**6257**	12,673 alerts	**1433**	18,782 alerts

The first portion of the graph represents the top 10 source addresses. As you can see from the graph, after the first two the number of hits dips significantly. Below are the top two.

```
May 16 00:02:20 MY.NET.196.193:1171 > 216.154.194.38:17300 SYN
******S*
May 16 00:02:20 MY.NET.196.193:1168 > 216.154.194.35:17300 SYN ******S*
May 16 00:02:20 MY.NET.196.193:1181 > 216.154.194.48:17300 SYN
******S*
```

Looking at the events generated by the first source, you can see a series of SYN attempts to port 17300. This looks like MY.NET.196.193 is browsing for external machines looking for the Kuang 2 trojan. This source definitely needs to be investigated and removed from the network. This type of traffic should also be blocked at the outer firewall.

```
May 18 17:47:15 MY.NET.87.50:27022 > 68.68.42.141:27005 UDP
May 18 17:47:15 MY.NET.87.50:27022 > 65.70.30.236:43620 UDP
May 18 17:47:15 MY.NET.87.50:27022 > 24.43.149.252:43620 UDP
```

Like the first one, the number two source is generating a huge amount of UDP traffic. A UDP Scan is another way for an attacker to see which hosts are active. There are a few tools attackers can use to perform this type of scan; UDP Scan and Nmap are two of them.

The second portion of the graph represents the top 10 destination addresses. From the graph, we can see that after the first host, the number of hits generated is almost negligible.

```
May 19 13:02:03 61.99.17.156:3760 > MY.NET.196.29:6555 UDP
May 19 13:02:04 61.99.17.156:3827 > MY.NET.196.29:2777 UDP
May 19 13:02:04 61.99.17.156:3915 > MY.NET.196.29:17698 UDP
```

Like the source addresses, the top destination address generated the bulk of the UDP Scans. Of the total 1,475,162 total scans detected, the UDP and SYN scans generated 1,458,674. The two scans generated over 98.89% of all the scans detected.

The second chart shows the top 10 source and destination ports. Below is a breakdown of what they are.

Rank	Source Port	Description	Target Port	Description
1	27022	Half-Life Game Server	17300	Kuang 2 trojan
2	7674	iMQ SSL tunnel	80	HTTP
3	22321	Unassigned	137	NetBIOS

4	1025	network blackjack	27005	FLEX LM (1-10)
5	1027	ExoSee	445	Microsoft-DS
6	2328	Netrix SFTM	23000	CVMMON
7	1026	Calender Access Protocol	7674	iMQ SSL tunnel
8	1028	Unassigned	21	FTP
9	1029	Unassigned	22321	Unassigned
10	6257	WinMX (old protocol)	1433	Microsoft SQL Server

From the chart above you see that the majority of traffic originated from port 27022. I did a search on The Internet Ports Database [33] and discovered that this port is largely associated with the Half-Life Game Server. It also listens for UDP traffic. Searching through SecurityFocus I did find two vulnerabilities associated with this. The first problem occurs through a machine connected to the Half-life server. The rcon command of the Half-life Linux Dedicated Server calls a function which contains an unchecked buffer. In this scenario, malicious user can bring up the game command console to execute commands, similar to that of an IRC server console, and send an rcon command to the server with enough data to overwrite the return address, causing the server to crash. [34]

The second problem consisted of a format string vulnerability. A function within rcon does not validate the input to the rcon command buffer, which is passed to sprintf() function. Therefore, it is possible for a malicious user to pass a specially formatted string via the rcon command that may result in remote code execution. [34]

The next table indicates that the majority of traffic was destined for port 17300. I did another search on The Internet Ports Database [33] and discovered that port 17300 has been linked to the Kuang 2 trojan. Searching through the data I found the majority of this traffic was being generated by MY.NET.196.193 and directed towards the 216.155.x.x and the 216.156.x.x networks. This would indicate to me that MY.NET.196.193 might be infected by the Kuang 2 trojan.

OOS (Out of Specification) Top Talkers

Below are the Top Ten Source and Destination Hosts based on the number of alerts generated from the **OOS log**. Sawmill was used for this analysis.

Rank	Total # Target Hits	Target IP	Total # Source Hits	Source IP
rank #1	579 alerts	**MY.NET.224.134**	970 alerts	**66.117.30.14**
rank #2	485 alerts	**MY.NET.24.22**	373 alerts	**81.57.90.18**
rank #3	477 alerts	**MY.NET.6.40**	316 alerts	**210.253.206.180**
rank #4	468 alerts	**MY.NET.6.47**	250 alerts	**209.123.49.137**
rank #5	453 alerts	**MY.NET.24.21**	249 alerts	**212.202.170.228**

rank #6	447 alerts	**MY.NET.211.26**	184 alerts	**213.186.35.9**
rank #7	437 alerts	**MY.NET.24.23**	179 alerts	**212.186.78.246**
rank #8	316 alerts	**MY.NET.237.118**	148 alerts	**213.197.11.147**
rank #9	282 alerts	**MY.NET.233.78**	145 alerts	**196.26.86.133**
rank #10	261 alerts	**MY.NET.24.44**	144 alerts	**209.47.197.17**

OOS Source # 1:
66.117.30.14

OOS Log:
05/17-13:13:43.327653 66.117.30.14:45010 -> MY.NET.219.198:1182
05/17-13:17:34.682517 66.117.30.14:54893 -> MY.NET.211.142:1182

Alert Log:
05/18-02:31:39.835239 [**] Queso fingerprint [**] 66.117.30.14:49416 -> MY.NET.233.78:1182
05/18-02:28:10.453523 [**] Queso fingerprint [**] 66.117.30.14:43981 -> MY.NET.224.134:1182

Scan Log:
May 18 02:31:39 66.117.30.14:49416 -> MY.NET.233.78:1182 SYN 12****S* RESERVEDBITS
May 18 02:28:10 66.117.30.14:43981 -> MY.NET.224.134:1182 SYN 12****S* RESERVEDBITS

OOS Source 1 appears to be sending out malformed packets by setting the ECN bits in the TCP header (**12******S*) on the initial SYN. Normally these bits are set to perform network congestion checks. This source is more than likely using a network mapping tool called Queso to perform network reconnaissance.

OOS Source # 2:
81.57.90.18

OOS Log:
05/17-15:31:27.761485 81.57.90.18:52574 -> MY.NET.218.154:6346
05/17-15:31:35.709444 81.57.90.18:52801 -> MY.NET.218.154:6346

Alert Log:
05/18-13:17:11.972995 [**] Queso fingerprint [**] 81.57.90.18:33301 -> MY.NET.218.154:6346
05/18-14:16:05.232576 [**] Queso fingerprint [**] 81.57.90.18:51279 -> MY.NET.185.48:6346

Scan Log:
May 18 11:41:40 81.57.90.18:40759 -> MY.NET.218.154:6346 SYN 12****S* RESERVEDBITS
May 18 13:16:50 81.57.90.18:33035 -> MY.NET.218.154:6346 SYN 12****S* RESERVEDBITS

OOS Source 2, at first glance, appears to be generating the same traffic as Source 1. But a close look at the target port (6346) leads me to believe this is not Queso again, but probably some P2P software like Gnutella or Bearshare that is sending out malformed packets.

OOS Source # 3:
210.253.206.180

OOS Log:
05/19-13:18:22.087974 210.253.206.180:53410 -> MY.NET.211.26:6011
05/19-14:56:10.220531 210.253.206.180:56014 -> MY.NET.211.26:6011

Alert Log:
05/18-07:31:47.743924 [**] Queso fingerprint [**] 210.253.206.180:54435 -> MY.NET.211.26:6011
05/18-07:53:08.068238 [**] Queso fingerprint [**] 210.253.206.180:55156 -> MY.NET.211.26:6011

Scan Log:
May 18 07:31:47 210.253.206.180:54435 -> MY.NET.211.26:6011 SYN 12****S* RESERVEDBITS
May 18 07:53:08 210.253.206.180:55156 -> MY.NET.211.26:6011 SYN 12****S* RESERVEDBITS

Here again, this looks very similar to Source 2 and exactly like Source 1. I'm starting to see a trend. The only difference appears to be the target ports. Port 6346, as in Source 2, was probably a random port chosen by Queso and it's just a coincidence that it's the same as the P2P ports.

OOS Source #4:
209.123.49.137

OOS Log:
05/16-07:02:37.322336 209.123.49.137:43511 -> MY.NET.220.14:6883
05/16-07:03:48.280347 209.123.49.137:45707 -> MY.NET.195.155:6887

Alert Log:
05/16-06:46:25.114277 [**] Queso fingerprint [**] 209.123.49.137:42234 -> MY.NET.220.14:6889
05/16-07:01:18.638493 [**] Queso fingerprint [**] 209.123.49.137:40672 -> MY.NET.194.35:6882

Scan Log:
May 16 07:01:18 209.123.49.137:40672 -> MY.NET.194.35:6882 SYN 12****S* RESERVEDBITS
May 16 06:29:36 209.123.49.137:37815 -> MY.NET.220.14:6886 SYN 12****S* RESERVEDBITS

The trend continues! I'm about 95% sure these are all related to Queso. I did see some other well known ports that had triggered in the OOS file. I'll check one more source to verify.

OOS Source #5:
212.202.170.228

OOS Log:
05/19-20:42:04.680409 212.202.170.228:34199 -> MY.NET.237.118:4662
05/19-20:42:04.680557 212.202.170.228:34198 -> MY.NET.237.118:4662

Alert Log:
05/19-20:20:52.327349 [**] Queso fingerprint [**] 212.202.170.228:59624 -> MY.NET.237.118:4662
05/19-21:02:08.184152 [**] Queso fingerprint [**] 212.202.170.228:37131 -> MY.NET.237.118:4662

Scan Log:
May 17 22:19:32 212.202.170.228:44766 -> MY.NET.237.118:4662 SYN 12****S* RESERVEDBITS

May 18 05:44:55 212.202.170.228:53153 -> MY.NET.237.118:4662 SYN 12****S* RESERVEDBITS

I'm convinced. All the packets are the same except for the target port. Let see what the Target hosts look like.

OOS Target #1:
MY.NET.224.134

OOS Log:
05/18-03:31:36.099221 66.117.30.14:37117 -> MY.NET.224.134:1182
05/18-03:32:30.582602 66.117.30.14:39904 -> MY.NET.224.134:1182

Alert Log:
05/17-13:59:05.350223 [**] Queso fingerprint [**] 66.117.30.14:47447 -> MY.NET.224.134:1182
05/17-17:07:30.445282 [**] Queso fingerprint [**] 66.117.30.14:39247 -> MY.NET.224.134:1182

Scan Log:
There are no entries in the scan log for this IP address.

I am fairly certain that the majority of events in the OOS logs, if not all of them, are the result of external sources performing scans on the MY.NET network. Predominantly, the tool of choice appears to be Queso, but any number of the other vulnerability and port scanning tools will exhibit this same behavior.

Defensive Recommendations

This analysis was put together with the intent of providing an overview of the University's network. From this report, the University will be able to better understand and react to the issues presently affecting them.

After a thorough analysis of the University's logs, it is clear that the University has little to no security in place. The most obvious examples of this are the numerous Nimda infections running ramped and the detection of Trojan activity within the environment. Serious resources are needed to help combat these out-of-control activities.

The first thing the University needs to do is to get a handle on the Nimda infected servers. The most effective way to do this is to incorporate an enterprise wide distribution of an Anti-Virus solution. This will be a good first step as it will protect those system not already infected. Which Anti-Virus product to use will depend on who-ever the University has the best professional relationship with. Symantec and McAfee are probably two of the biggest, but any of them will do. One caveat to remember is that due to the behavior of Nimda and the holes it leaves behind, the only certain way to clean it is to rebuild it. This is the recommended approach. If the server is a mission critical server that cannot be taken offline, then all measures need to be taken to ensure the system is property cleaned and patched. There are numerous documents and tools available to assist the administrators in this process.

The next phase of remediation should be focused on the Trojan activity. There appears to be numerous servers infected either via the IRC SDCC backdoor of from SubSeven. The

Anti-Virus solution recommended about will definitely help out in this space. In addition, these particular servers will need to be tracked down, cleaned, patched or rebuilt to ensure that the holes created are plugged up.

By this time the University staff should be making some good progress in cleaning up the environment of known infectors. Now, more attention will need to be given to the constant noise makers that are making the environment difficult to protect. To do this, the University will need to track down all the P2P and gaming systems. It was noted above that there is a lot of traffic being generated by these types of applications. In addition to the P2P and gaming systems, there are also a lot of scanning activities taking place. Some of this is related to the P2P and gaming applications, but there is also a lot of legit scanning taking place. To fully complete this phase, the University will need to complete some other tasks besides just removing the software.
They will need to create, publish, and enforce security policies in addition to providing security training. Basically, they need to push security awareness to all staff and users.

Finally, as part of an initial layer of defense from external attacks as well as internal ones, the University should implement Egress and Ingress filtering on their routers and firewalls. This will enable them to quickly cut down on the number of scans they're seeing in their environment. Also, ACL's should also be implemented to assist in blocking all unwanted traffic, such as the SYN flood attacks and UDP scans noted above. Having these measures in place will also assist in eliminating the P2P and gaming application because you can block the common ports associated with these applications. There are also numerous other benefits to having these measures in place besides the ones described above.

There have been a lot of recommendations laid out so far. All of them are crucial to the security of the University's network. These changes will not happen overnight, but a consistent effort will need to be made if this transition is to be successful. Some of these recommendations might not be acceptable for an Educational facility, but the University should attempt to apply all changes relevant to their policies. Failure to do so will result in a lapse of security and may make the University liable for not performing their due diligence.

Description of the Analysis Process

The first step in the analysis process is the gathering of the data. Per SANS requirements I had to download five consecutive days worth of data that is posted on the incidents.org [10] web site. The five days that I chose were 5/16/03 – 5/20/03. Each day has three different files that need to be downloaded. There are Scans files, Alert files, and OOS files. Each file serves its own purpose and all of them need to be analyzed to show the different trends, events and correlation. I used a variety of tools to perform the analysis. Two of the main tools used were SnortSnarf [9] and Sawmill [5]. These tools, and others, were used to generate tables and reports in a usable, HTML format that made it easier to spot suspicious traffic and anomalies.

Before I could use any of the programs available, I had to do some manipulation of the log files. This would prove to make my life a lot easier. One of the first things I did with each group of logs was to concatenate them together in order of date. This gave me three master files; one for each group. I used a combination of Unix commands to assist me with this step. Some of those commands were: grep, cat, sed, cut and wc.

At this time I now have three master files that need to be analyzed. The main file for my analysis was the alert file. This file was quite large. I took Tod Beardsley's [1] advice and removed all the entries that were generated by the spp_portscan preprocessor because these events were shown to be caused by peer-to-peer file sharing application. I also took Tod's advice and changed all the instances of "MY.NET" with a numeric value that was not in the concatenated alert file. After many grep attempts, I finally found a suitable combination. I replaced the "MY.NET" with "192.182". With the new combination, I was able to run a successful SnortSnarf against it.

With the SnortSnarf output, it was easier to find events of interest. This also gave me a better understanding of what was occurring on the University's network. I was also able to focus my attention on looking for the specific addresses, ports or events from the scans and OOS files. To assist me with this, I used a combination of grep and a program called Sawmill. From the output of Sawmill, I was able to focus my grep statements on more specific criteria. With all this data presently at hand, I was able to perform a detailed analysis of the University's network.

References

[1] Beardsley, Tod. "GCIA Certification – Practical Assignment" URL: http://www.giac.org/practical/Tod_Beardsley_GCIA.doc (May 24, 2003).

[2] CERT.org. "CERT® Advisory CA-2001-13 Buffer Overflow In IIS Indexing Service DLL" URL: http://www.cert.org/advisories/CA-2001-13.html (May 31, 2003).

[3] CERT.org. "CERT® Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL" URL: http://www.cert.org/advisories/CA-2001-19.html (May 31, 2003).

[4] CERT.org. "CERT® Advisory CA-2001-26 Nimda Worm" URL: http://www.cert.org/advisories/CA-2001-26.html (June 8, 2003).

[5] Sawmill.net. "Welcome to Sawmill" URL: http://www.sawmill.net (May 24, 2003).

[6] Sourcefire, Inc. "Snort Signature Search" URL: http://www.snort.org/ (May 31, 2003).

[7] Snort_stat. Chen, Yen Ming. "Snort_stat." URL: http://www.snort.org/dl/contrib/data_analysis/snort_stat.pl (May 24, 2003).

[8] Snortalog. Chartier, Jeremy. “SNORTALOG: SNORT Analyser Logs.” URL: http://www.snort.org/dl/contrib/data_analysis/snortalog (May 24, 2003).

[9] SnortSnarf. “SnortSnarf Snort alert browser.” URL: http://www.silicondefense.com/software/snortsnarf/index.htm (May 24, 2003).

[10] Incidents.org “GCIA Practical Logs, Assignment 3.” URL: http://www.incidents.org/logs (June 14, 2003).

[11] Alexander, Bryce. “Intrusion Detection FAQ – Port 137 Scan.” URL: http://www.sans.org/resources/idfaq/port_137.php (June 14, 2003).

[12] Singer, Abe. “Analysis of network.vbs worm.” URL: http://security.sdsc.edu/publications/network.vbs.shtml (June 14, 2003).

[13] Grout, Chris. “GCIA Certification – Practical Assignment” URL: http://www.giac.org/practical/Chris_Grout.doc (June 14, 2003).

[14] Embrich, Mark. “GCIA Certification – Practical Assignment” URL: http://www.giac.org/practical/Mark_Embrich_GCIA.htm (June 14, 2003).

[15] Roesch, Marty. “Tiny Fragments; Neohapsis Archives” URL: http://archives.neohapsis.com/archives/snort/2000-05/0103.html (June 14, 2003).

[16] Wu, Marcus. “GCIA Certification – Practical Assignment” URL: http://www.giac.org/practical/GCIA/Marcus_Wu_GCIA.pdf (June 14, 2003).

[17] oWn3d. “IRC FAQ» What's an XDCC?” URL: http://www.dslreports.com/faq/4493 (June 14, 2003).

[18] Ellis, Joe. “GCIA Certification – Practical Assignment” URL: http://www.giac.org/practical/Joe_Ellis_GCIA.doc (June 14, 2003).

[19] Hawrylkiw, Dan. “GCIA Certification – Practical Assignment” URL: http://www.giac.org/practical/Dan_Hawrylkiw_GCIA.doc (June 14, 2003).

[20] Menke, Mark. “GCIA Certification – Practical Assignment” URL: http://www.giac.org/practical/Mark_Menke_GCIA.doc (June 17, 2003).

[21] Common Vulnerabilities and Exposures. “CVE-2001-0906” URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0906 (June 17, 2003).

[22] Insecure.org. “Nmap” URL: http://www.insecure.org/nmap/ (June 21, 2003).

[23] Credeur, Brian. “GCIA Certification – Practical Assignment” URL: http://www.giac.org/practical/Brian_Credeur_GCIA.doc (June 14, 2003).

[24] Stewart, Joe. “Neohapsis Archives” URL:
http://archives.neohapsis.com/archives/snort/2000-11/0244.html (June 21, 2003).

[25] arachNIDS. “IDS29 Probe-Queso Fingerprint Attempt” URL:
http://whitehats.com/cgi/arachNIDS/Show?_id=ids29&view=event (June 21, 2003).

[26] Holstein, Michael. “GCIA Certification – Practical Assignment” URL:
http://www.giac.org/practical/Michael_Holstein_GCIA.doc (June 14, 2003).

[27] Snort.org. “SHELLCODE x86 NOOP” URL:
http://www.snort.org/snort-db/sid.html?sid=648 (June 21, 2003).

[28] Snort.org. “SHELLCODE x86 stealth noop” URL:
http://www.snort.org/snort-db/sid.html?sid=651 (June 21, 2003).

[29] So, Hee. “GCIA Certification – Practical Assignment” URL:
http://www.giac.org/practical/Hee_So_GCIA.doc (June 14, 2003).

[30] Larrieu, Heather, “GCIA Certification – Practical Assignment” URL:
http://www.giac.org/practical/GCIA/Heather_Larrieu_GCIA.doc (June 22, 2003).

[31] Snort.org. “SHELLCODE x86 setuid 0” URL:
http://www.snort.org/snort-db/sid.html?sid=650 (June 22, 2003).

[32] Snort.org “Writing Rules – Section 2.3.15” URL:
http://www.snort.org/docs/writing_rules/chap2.html#tth_sEc2.3.15 (June 22, 2003).

[33] The PortsDB Project. “The Internet Ports Database” URL:
http://www.portsdb.org/bin/portsdb.cgi (June 22, 2003).

[34] SecurityFocus. “BID 1847 – Halflife Linux Server rcon Vulnerabilities” URL:
http://www.securityfocus.com/bid/1847 (June 22, 2003).