Global Information Assurance Certification Paper

Intrusion Prevention Systems:
The New Frontier of Intrusion Detection?

Ron Shuck, CISSP
GIAC GCIA Practical (version 3.3)
SANS Network Security, Washington, DC, USA
October 19-24, 2002
Submitted: April 1, 2003

Table of Contents

Assignment 1: The State of Intrusion Detection

Intrusion Prevention Systems: The New Frontier of Intrusion Detection?

Summary:

What is an Intrusion Prevention System? In simple terms, one would say an Intrusion Prevention System or IPS is a system that prevents an intrusion. Wait a minute, is a good Security Policy an IPS? How about a Firewall, or Antivirus, or even a VPN? If preventing an intrusion is all that is required, isn't virtually every security tool an IPS?

The primary goal of this paper is to clearly define the term Intrusion Prevention System with out all of the hype and marketing doubletalk, and explain why Intrusion Detection and Intrusion Prevention systems are two separate types of security products. However, this is not a strictly black and white issue. Ultimately you have to decide.

History:

The term first showed up over two years ago when it was used by a company called Click-Net for their new product Entercept. Since then the company has changed their name to Entercept.[1] Entercept now claims to be the proven leader in intrusion prevention. Since then, countless vendors have hopped on the "intrusion prevention" bandwagon. Each vendor uses the term to identify different products, but the bottom line is that the phrase could mean anything.

Definition of IDS:

Most can agree that an Intrusion Detection System is any system used to detect intrusions or attacks. This can also be expanded to include protocol and transport anomalies, or changes in normal behavior. These systems are typically separated into two distinct classes: Network-based Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (HIDS).

Method of Detection

Regardless of the type of IDS, both have common methods for determining or detecting attacks and anomalies. The most common is signature-based. Signature based systems work by matching events or network traffic to a pre-determined list of "signatures". The signature is a description of the attack, such as network characteristics or system activities. There are also new hybrid systems in development that will utilize fuzzy logic and data mining in conjunction with traditional signatures to detect attacks. There are even plans for systems that will "learn" about new attacks using Autonomous Reinforcement Learning.

[1] Briney, URL: http://www.infosecuritymag.com/2002/apr/note.shtml

Network-based IDS

A Network-based IDS is typically a passive device that monitors traffic on a network segment, and generates alerts or logs when an attack or intrusion is detected. Later, we will see that new technology is challenging this paradigm, but let's stick with this definition for now. Below are some common examples of network based products. An excellent resource for locating Open Source IDS solutions can be found at:
http://www.whitehats.com/index.shtml

Snort

Written by Martin Roesch, Snort is a very powerful Open Source based network intrusion detection system capable of real-time traffic and protocol analysis and content matching. Snort is a major player in the NIDS market. The Snort system utilizes signatures to determine attacks. This is not only research information. I have deployed several Snort systems, and can attest to the effectiveness of the Snort system. Snort can be found at: http://www.snort.org/
Commercial IDS solutions by the creators of Snort are also available from Sourcefire located at: http://www.sourcefire.com/

ISS Real Secure

Produced by Internet Security Systems (ISS), the Real Secure product line offers a full featured network intrusion detection system. The Real Secure product contains many of the features desired in a NIDS. It was my experience that the missing component was the detailed view or display of the packet that caused the alert. Real Secure is also signature driven. The Real Secure product line also incorporates a host based system. The major advantage of Real Secure is the product integration. ISS has a system that can integrate network, host, and desktop intrusion detection, as well as vulnerability scanners into a single, centrally managed system. ISS Real Secure can be found at:
http://www.iss.net/products_services/enterprise_protection/

Cisco Secure IDS (Netranger)

Originally developed by The Wheel Group and called "NetRanger", the Cisco Secure IDS sensor is now a NIDS hardware appliance. The real advantage to the Cisco solution is network integration. The Secure IDS systems are available in a variety of deployment options. They can be stand–alone sensors or blades in the 6500 Chassis series switches. They can also be integrated as a software component into existing Cisco routers and Firewalls. Cisco also offers a host based solution as well. The Cisco Secure IDS can be found at: http://www.cisco.com/

SHADOW

No discussion of network-based intrusion detection would be complete with out the mention of SHADOW. The project was developed by Stephen Northcutt and the Shadow team back in 1994 for the Naval Surface Warfare Center. Shadow stands for Secondary Heuristic Analysis for Defensive Online Warfare, and is still in use today.
Shadow can be found at: http://www.nswc.navy.mil/ISSEC/CID/index.html

Host-based IDS

A Host-based IDS, is typically software that resides on the host system and detects attacks or intrusions for that system. These systems can be strictly detection systems such as Tripwire, or they can restrict events to "trusted" or specified machines like the TCP Wrappers software. Again, emerging technology is changing the face of host based systems. Later, we will explore some of these new approaches. Below are some common examples of host based products.

TCP Wrappers

Written by Wietse Venema, TCP Wrappers allows for logging and access control of some common services such as exec, ftp, rsh, telnet, rlogin, finger, etc. TCP Wrappers can be found at: ftp://ftp.porcupine.org/pub/security/

Xinetd

Xinetd, which stands for eXtended InterNET services daemon, is a replacement for the traditional inetd. The internet services daemon controls many of the common services on a "*nix" based system. Xinetd provides enhancements over TCP Wrappers. Xinetd can be found at: http://www.xinetd.org/

Tripwire

Written by Eugene Spafford and Gene Kim, Tripwire is basically software that records information on the files you specify, and can then notify if one of those files has been changed. Tripwire can be found at: http://www.tripwire.com/

Swatch

Swatch was written by Todd Atkins, and basically is a log file analyzer. It can notify on certain events, or even take action based on an event in the log file. Swatch can be found at: http://swatch.sourceforge.net/

Manual Audit

This is the good old fashion method for intrusion detection. It involves manually reviewing the various log files, a baseline of open ports, and the processes running on the system. Although, we may cringe at the thought of manually reviewing log files, etc., this is the tried and true mechanism for host based security. In fact, many products and HIDS applications are based on these principles. A well administered system is the first line of defense in host based security.

Various Claims of Intrusion Prevention

Even if you are new to IDS technology, by now you have a basic understanding of what constitutes an Intrusion Detection System. An IDS detects intrusions, simple enough. So, you may think that an Intrusion Prevention System would be the next evolution of IDS, right? Wrong! "Intrusion detection and intrusion prevention aren't different names for the same market segment—they're different names for two distinct categories of security products."[2]

[2] Taylor, URL: http://www.zdnet.com.au/itmanager/technology/story/0,2000029587,20267597,00.htm

For the sake of argument, let's assume that you are satisfied that Intrusion Detection and Intrusion Prevention are different products. Then what is Intrusion Prevention? If you search the Internet for "intrusion prevention system", you will get thousands of matches. When I initiated this search, I was presented with about 95,700 matches. Good grief! How can a security professional ever review that much information? There are countless vendors with products that offer Intrusion Prevention Systems. They have cool and impressive sounding features like Dynamic Attack Suppression™ and Integrated Attack Mitigation, Automatic Policy Generation, and Intrusion Prevention Ecosystem. The following section will skip all the marketing hype, and describe the basics of Intrusion Prevention.

Types of Intrusion Prevention Systems

I think we can all agree that in its simplest form an Intrusion Prevention System should prevent intrusions. That seems easy enough. Regardless of the hype, cool terms and phrases, all Intrusion Prevention Systems provide this "prevention" in one of three ways. The system either stops the intrusion at the Operating System (OS) level, at the application level, or at the network level. With this in mind, let's review the various methods used by prevention systems to stop intrusions.

Trusted Operating System

The idea of a Trusted OS is that the entire operating system is secure. There are really two varieties; first is the trusted version of a vendor OS such as "Trusted Solaris" from Sun or HP's "VirtualVault." These are operating systems with built-in security. Second is a third party application or OS Wrapper. These applications generally replace the kernel to provide security for the OS. Examples of this type of trusted OS are PitBull LX by Argus Systems and Linux Lockbox by Guardian Digital.

In either case, the Trusted OS provides these major security features:

- Compartmentalization of resources such as files or processes. This provides security by allowing access only to the files or resources that are appropriate for a given function.
- Compartmentalization of user roles. This provides security by restricting what actions each user has the ability to perform including the administrator.
- Enforcement of least privilege. This provides security by only allowing the actions that are required.
- Kernel level enforcement. This provides security by implementing the security decisions at a low level closer to the resource being protected.
- Sensitivity labeling of files and resources. This provides security by enforcing mandatory access controls as opposed to discretionary.

™ Dynamic Attack Suppression is a Trademark of Latis Networks, Inc.

Host Intrusion Prevention (HIP)

These systems are very similar to a Trusted OS, and may even provide many of the same security features. Typically they are developed to protect one or more specific applications. These systems will generally utilize kernel level modules or replace shared libraries. The focus being the interception of system calls to look for bad application behavior. HIP systems are implemented in three basic methods: behavior-based policies, signature-based policies, and user-based policies.

Some common examples of HIP systems are Entercept Advanced e-Server or Standard Edition by Entercept Security Technologies, STAT Neutralizer™ by Harris, Real Secure® Server Sensor from ISS, and Okena's Stormwatch™. There are also many products that are focused on specific types of servers such as Entercept's Web Server and Database Editions and SecureIIS™ Web Server Protection from eEye Digital.

Application Firewall

An Application Firewall is somewhat less intrusive than a Trusted OS or HIP. Where a Trusted OS or HIP attempts to protect the server or host, the Application Firewall protects specific applications. Application Firewalls are known by many different names: Content Scrubbers, Proxy and Proxy Firewall, Reverse Proxy, and Air Gap. Although, there are different types of Application Firewalls, most are dedicated to securing common inbound traffic such as web and e-mail traffic.

Content Scrubbers

Content scrubbers perform just as their name implies. They remove "malicious" traffic or content. These devices typically sit in-between the Firewall and some content server such as web, mail, etc.

Proxy

A Proxy acts much like the definition, an agent or substitute. A proxy acts on behalf of the client. So, the client talks to the proxy and the proxy talks to the server. Proxies are often referred to as Application Proxies or Application Layer Firewalls. The Proxy actually has to re-create the packet to send to the server. This allows the proxy to not only record the request, but inspect the traffic. The most common examples of proxies are Microsoft's Proxy and ISA Servers, and the Linux proxy Squid.

Reverse Proxy

A Reverse Proxy is very similar to a regular or forward proxy. While a forward proxy acts on behalf of the client, the reverse proxy acts on behalf of the server. This is typically used to provide access to one or more servers behind a Firewall. The reverse proxy would sit outside the Firewall or in a DMZ, and then proxy connections from external users to the internal server or servers. The Firewall would be configured to only allow the reverse proxy to access the internal servers. Common examples are Netscape Proxy Server and Apache Web Server using 'mod_proxy'.

™ STAT Neutralizer is a Trademark of Harris Corporation
® Real Secure is a registered trademark of Internet Security Systems
™ Stormwatch is a Trademark of Okena, Inc.
™ SecureIIS is a Trademark of eEye Digital Security

Air-Gap

The Air-Gap is similar to a reverse proxy. Both air-gap and reverse proxy are application specific. This means the device or proxy has to be designed for each different application or protocol. The air-gap is different in the way it gets data from the source to the destination. These devices use one or more gap technologies like "real time switch", "one way link", or "network switcher". The basic idea is that the device contains two independent components: one for the trusted and one for the un-trusted networks. They use a separate hardware device, such as a SCSI bus, to transfer data only from one to the other. Two common examples are Whale Communications e-Gap and Spearhead's AirGAP.

Gateway or In-line IDS

These devices are much like a typical IDS. The difference is that these devices are not passive. All traffic must pass through the in-line IDS. This means that the IDS can drop packets just like a Firewall. A common example would be ISS's Real Secure® Guard.

Passive or Active Response

This involves extending the functionality of standard intrusion detection systems. Many IDS systems have the "passive response" capability. This involves the IDS sensor crafting a TCP RST packet in response to a specific alert or rule. The problem is that this is not always reliable. It depends on the type of exploit (it may already be too late), and the timing because the intended destination is going to respond to the "bad" packet as well.

The other type of response is active response. This usually involves the IDS sensor communicating with a Firewall to change rules "on-the-fly". For example, an IDS receives a "bad" packet from an IP, it can insert a new Firewall rule that blocks traffic from that address. The new Firewall rule can be very specific or very general depending on the implementation. StillSecure's™ BorderGuard is a prime example. This product uses Snort as the IDS and can work with various Firewalls to block traffic. The coordination with the Firewall can be via a downloadable ACL for Cisco's PIX, or via Open Platform for Security (OPSEC) for Check Point Firewall-1/VPN-1 and other OPSEC compliant providers™.

Security Analysis

Finally, there are the analysis tools. These tools generally consolidate and correlate data from Firewall logs, IDS logs, and other sources. This data is then analyzed to locate not only intrusions but other types of problems. Sometimes these tools are called Security Information Management Systems or SIMS. Two prime examples are SilentRunner® and netForensics®.

® Real Secure is a registered trademark of Internet Security Systems
™ StillSecure is a Trademark of Latis Networks
™ OPSEC, VPN-1 and Firewall-1 are Tradmarks of Check Point Software Technologies Ltd.
® SilentRunner is a registered trademark of Silentrunner, Inc.
® netForensics is a registered trademark of Netforensics, Inc.

Conclusion

Hopefully, you now have a better understanding of the primary Intrusion Detection Systems and some of the Intrusion Prevention Systems. It should be apparent that Intrusion Prevention is much more like a Firewall than an IDS. It should also be obvious that there is a lot of hype regarding Intrusion Prevention. I think this sums it up nicely. "The point is that if "intrusion prevention" can refer to everything, it can't mean anything-that is, it can't mean any one thing. It's a convenient marketing neologism designed to make you think it's the next evolution in IDSes."[3]

So, security professional beware! Look through the hype at what the product can actually accomplish. Know and understand what you want a security device to do before you unwrap your shiny, new, IPS. Remember, "IDS and IPS are not competing devices."[4] You as the security professional are the greatest intrusion prevention system. One of the IDS Titians, Stephen Northcutt, said in his book on the subject, "… there is no magic product that can do intrusion detection for you, in the end, every analyst needs a basic understanding of how IP works, so they will be able to detect the anomalies."[5]

If you are still unconvinced, and you still do not believe. I went straight to the source. My all time IDS hero and coolest famous guy I have ever met. He had this to say.

> "I think that network IPS is the future of firewalling, not of intrusion detection. There are a lot of people who will tell you that IPS will replace the IDS, but that ignores the fact that they serve two separate functions. An IPS is an access control device, an IDS is a network monitoring tool. One provides protection, the other provides awareness. IPS won't do anything to monitor internal host-to-host communications, it certainly won't tell you when attacks take place on those links. Intrusion detection also provides validation that your IPS is working or not. Let's face it, the only way to tell if you IPS has failed and let an attack through is with an IDS."[6]

[3] Briney, URL: http://www.infosecuritymag.com/2002/apr/note.shtml
[4] Synder, p.22.
[5] Northcutt, p.xiii
[6] Roesch, E-mail

References:

Briney, Andy. “What Isn't Intrusion Prevention?” April 2002.
URL: http://www.infosecuritymag.com/2002/apr/note.shtml (26 Jan. 2003).

Synder, Joel. “Intrusion Prevention Essentials.” SANS Institute Webcast Slide Presentation. 4 Dec. 2002.

Bridges, Susan, and Rayford Vaughn. “Fuzzy Data Mining and Genetic Algorithms Applied To Intrusion Detection.”
URL: http://csrc.nist.gov/nissc/2000/proceedings/papers/005.pdf (16 Feb. 2003).

Cannady, James. “Next Generation Intrusion Detection: Autonomous Reinforcement Learning.”
URL: http://csrc.nist.gov/nissc/2000/proceedings/papers/033.pdf (16 Feb. 2003).

Goldman, Jeff. “Intrusion Detection Systems: SHADOW.” May 2002.
URL: http://www.isp-planet.com/services/ids/shadow.html (22 Feb. 2003).

Fratto, Mike. “Keep Out.” October 2002.
URL: http://www.nwc.com/1322/1322f1.html?ls=TW_110402_rev (22 Feb. 2003)

Northcutt, Stephen, Judy Novak, and Donald McLachlan. Network Intrusion Detection – An Analyst’s Handbook Second Edition. Indianapolis: New Riders Publishing, 2000.

Piscitello, David. “Intrusion Detection…Or Prevention?” Business Communications Review. May 2002. URL: http://www.bcr.com/bcrmag/2002/05/p42.asp (22 Feb. 2003).

Fisher, Dennis. “New Wave of IDS Tools Take Aim at Prevention.” September 2002.
URL: http://www.eweek.com/print_article/0,3668,a=31080,00.asp (23 Feb. 2003).

Fisher, Dennis. “New IDS Tools Automate Response.” November 2002.
URL: http://www.eweek.com/article2/0,3959,685311,00.asp (23 Feb. 2003).

Messmer, Ellen. “Intrusion Prevention Raises Hopes, Concerns.” Network World. November 2002. URL: http://www.nwfusion.com/news/2002/1104prevention.html (23 Feb. 2003).

Taylor, Laura. “Intrusion Detection is not Intrusion Prevention.” August 2002.
URL: http://www.zdnet.com.au/itmanager/technology/story/0,2000029587,20267597,00.htm (23 Feb. 2003)

Halme, Lawrence, and R. Kenneth Bauer. “Intrusion Detection FAQ - AINT Misbehaving: A Taxonomy of Anti-Intrusion Techniques”
URL: http://www.sans.org/resources/idfaq/aint.php (16 Feb. 2003).

Rhodes, Brandon, James Mahaffey, and James Cannady. “Multiple Self-Organizing Maps for Intrusion Detection”
URL: http://csrc.nist.gov/nissc/2000/proceedings/papers/045.pdf (16 Feb. 2003).

Lindstrom, Pete. “Diverse security technologies deliver the same message: Keep Out!" Guide to Intrusion Prevention. October 2002.
URL: http://www.infosecuritymag.com/2002/oct/sidebar.shtml (23 Feb. 2003).

Jacobs, Charles. “Trusted Operating Systems.” May 2001.
URL: http://www.sans.org/rr/securitybasics/trusted_OS.php (23 Feb. 2003).

Roesch, Martin. “RE: Intrusion Prevention.” E-mail to the author. 19 Nov. 2002.

Sapiro, Benjamin. “Application Level Content Scrubbers” August 2001.
URL: http://www.sans.org/rr/firewall/scrubbers.php (31 Mar 2003).

Cabral, Jim. “Securing Email Through Proxies: Smap and Stunnel” September 2001.
URL: http://www.sans.org/rr/email/smap.php (31 Mar 2003).

Stricek, Art. “A Reverse Proxy Is A Proxy By Any Other Name” January 2002.
URL: http://www.sans.org/rr/web/reverse_proxy.php (31 Mar 2003).

Assignment 2: Network Detects

Detect 1: BACKDOOR Q access

This output is the standard log output from Snort, and has the following format;

```
Alert Signature Name
Date - Time  Source IP:Source Port  -> Destination IP:Destination Port
IP Header Information
Protocol Information

 [**] BACKDOOR Q access [**]
10/31-18:10:04.866507 255.255.255.255:31337 -> 207.166.253.145:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
10/31-18:56:25.966507 255.255.255.255:31337 -> 207.166.97.171:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
10/31-21:08:05.026507 255.255.255.255:31337 -> 207.166.93.139:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
10/31-21:11:13.996507 255.255.255.255:31337 -> 207.166.125.50:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
10/31-22:44:17.156507 255.255.255.255:31337 -> 207.166.10.242:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
10/31-23:01:50.156507 255.255.255.255:31337 -> 207.166.243.229:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
10/31-23:33:32.136507 255.255.255.255:31337 -> 207.166.163.228:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-00:00:32.216507 255.255.255.255:31337 -> 207.166.100.235:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-00:40:35.196507 255.255.255.255:31337 -> 207.166.148.62:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

```
 [**] BACKDOOR Q access [**]
11/01-01:59:56.246507 255.255.255.255:31337 -> 207.166.136.161:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-02:03:53.196507 255.255.255.255:31337 -> 207.166.84.0:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-02:21:29.266507 255.255.255.255:31337 -> 207.166.150.179:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-02:30:05.226507 255.255.255.255:31337 -> 207.166.105.163:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-03:43:08.396507 255.255.255.255:31337 -> 207.166.121.149:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-03:51:08.246507 255.255.255.255:31337 -> 207.166.75.164:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-05:14:20.316507 255.255.255.255:31337 -> 207.166.134.120:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-06:01:41.386507 255.255.255.255:31337 -> 207.166.49.222:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-06:01:44.406507 255.255.255.255:31337 -> 207.166.195.220:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-06:03:02.406507 255.255.255.255:31337 -> 207.166.108.206:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-06:41:56.386507 255.255.255.255:31337 -> 207.166.248.169:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

```
 [**] BACKDOOR Q access [**]
11/01-06:56:38.426507 255.255.255.255:31337 -> 207.166.204.81:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-08:00:50.426507 255.255.255.255:31337 -> 207.166.175.116:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-08:28:20.476507 255.255.255.255:31337 -> 207.166.216.118:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-09:03:44.506507 255.255.255.255:31337 -> 207.166.249.188:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 [**] BACKDOOR Q access [**]
11/01-09:35:02.556507 255.255.255.255:31337 -> 207.166.191.175:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
11/01-09:44:56.566507 255.255.255.255:31337 -> 207.166.140.164:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
11/01-12:03:05.656507 255.255.255.255:31337 -> 207.166.84.156:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
11/01-12:35:20.736507 255.255.255.255:31337 -> 207.166.71.180:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
11/01-12:59:59.666507 255.255.255.255:31337 -> 207.166.42.113:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
11/01-13:22:02.716507 255.255.255.255:31337 -> 207.166.166.69:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
11/01-13:34:47.786507 255.255.255.255:31337 -> 207.166.148.94:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

```
[**] BACKDOOR Q access [**]
11/01-16:41:49.906507 255.255.255.255:31337 -> 207.166.178.165:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
11/01-16:41:52.896507 255.255.255.255:31337 -> 207.166.198.211:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
[**] BACKDOOR Q access [**]
11/01-17:24:16.986507 255.255.255.255:31337 -> 207.166.143.84:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Source of Trace:

The source of the trace was the raw logs directory at Incidents.org. The log files are the result of a Snort instance running in binary logging mode. The logs themselves have been sanitized. The specific file used for this detect was;
http://www.incidents.org/logs/Raw/2002.10.1

Since the network layout was not provided, I can only assume, but there is strong evidence that the Snort sensor that captured the data was located between two Cisco devices. Not only the packets involved in this detect, but all packets from the above log file have one of two distinct MAC addresses. There are two different OUIs and both belong to Cisco. My guess would be that the two devices are an external router and a PIX firewall or NAT router.

```
11/01/2002 00:10:04.866507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.253.145.515: tcp 3
11/01/2002 00:56:25.966507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.97.171.515: tcp 3
11/01/2002 03:08:05.026507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.93.139.515: tcp 3
11/01/2002 03:11:13.996507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.125.50.515: tcp 3
11/01/2002 04:44:17.156507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.10.242.515: tcp 3
11/01/2002 05:01:50.156507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.243.229.515: tcp 3
11/01/2002 05:33:32.136507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.163.228.515: tcp 3
11/01/2002 06:00:32.216507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.100.235.515: tcp 3
11/01/2002 06:40:35.196507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.148.62.515: tcp 3
11/01/2002 07:59:56.246507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.136.161.515: tcp 3
11/01/2002 08:03:53.196507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.84.0.515: tcp 3
11/01/2002 08:21:29.266507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.150.179.515: tcp 3
```

```
11/01/2002 08:30:05.226507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.105.163.515: tcp 3
11/01/2002 09:43:08.396507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.121.149.515: tcp 3
11/01/2002 09:51:08.246507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.75.164.515: tcp 3
11/01/2002 11:14:20.316507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.134.120.515: tcp 3
11/01/2002 12:01:41.386507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.49.222.515: tcp 3
11/01/2002 12:01:44.406507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.195.220.515: tcp 3
11/01/2002 12:03:02.406507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.108.206.515: tcp 3
11/01/2002 12:41:56.386507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.248.169.515: tcp 3
11/01/2002 12:56:38.426507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.204.81.515: tcp 3
11/01/2002 14:00:50.426507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.175.116.515: tcp 3
11/01/2002 14:28:20.476507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.216.118.515: tcp 3
11/01/2002 15:03:44.506507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.249.188.515: tcp 3
11/01/2002 15:35:02.556507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.191.175.515: tcp 3
11/01/2002 15:44:56.566507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.140.164.515: tcp 3
11/01/2002 18:03:05.656507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.84.156.515: tcp 3
11/01/2002 18:35:20.736507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.71.180.515: tcp 3
11/01/2002 18:59:59.666507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.42.113.515: tcp 3
11/01/2002 19:22:02.716507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.166.69.515: tcp 3
11/01/2002 19:34:47.786507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.148.94.515: tcp 3
11/01/2002 22:41:49.906507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.178.165.515: tcp 3
11/01/2002 22:41:52.896507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.198.211.515: tcp 3
11/01/2002 23:24:16.986507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
255.255.255.255.31337 > 207.166.143.84.515: tcp 3
```

The other evidence is that all of the packets of this detect, and all of the packets in the log file, contain a bad IP header checksum. This was completed visually in Ethereal for the packets of the detect, and using the following commands for the entire log file.

```
# tcpdump –vnr 2002.10.1 | wc –l
# tcpdump –vnr 2002.10.1 | grep "bad cksum" | wc –l
```

Both commands returned the same result, 14597 records. This is evidence that the protected network addresses were "munged" or obfuscated as stated in the README file in the log directory at Incidents.org.

I also did a sorted visual scan of both source and destination addresses of the entire log file. I did not detect any private networks. This is in no way conclusive, but it leads me to believe that the sensor was located outside the firewall or NAT device, and inside an external router providing connectivity to the Internet.

Detect was generated by:

The detect was generated using Snort v 1.9.0 (Build 209) with a default "snort.conf" v1.110 and "backdoor.rules" v1.25 on Red Hat Linux 7.3. The detect was analyzed using Snort log files, ACID, Ethereal, Sniffer Pro, and Tcpdump.

The specific rule that triggered the alert was:

```
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;)
```

This rule basically alerts on any network packet that is TCP, has a source network of 255.255.255.0, the ACK flag set in addition to any other flags, and a packet payload size of 2 or more. Of course the source and destination ports can contain any value, and the destination address can be any address because the default configuration for $HOME_NET is "any".

Probability the source address was spoofed:

The probability that the source address was spoofed is 1 or 100%. These packets have strong evidence of crafting. First, a source address of "255.255.255.255" is not valid for normal IP traffic. It is possible that the packet was sent using an obsolete form of all-zero broadcast address (according to the "Expert mode' analysis in Sniffer Pro 4.60.01), but that is highly unlikely. Second, the sequence number is 0, which is possible, but again very unlikely since it was 0 in all of the detect packets. Third, the ACK and RST flags are both set which is only valid as a response to a SYN on a port that is not listening. Fourth, is that the source port is 31337 or the "eleet" port common in hacker tools. This is a valid ephemeral port, but not likely when combined with the other evidence. Finally, I question the TTL value. The smallest default TTL value is 32 which is used by Windows. So, it is possible that the packets originated 17 hops away from a Windows machine, but again combined with the other factors I would suspect packet crafting.

The following is a representative example of the detect packets. They were all identical to this packet with the exception of the destination IP and obvious checksum values.

```
11/01/2002 00:10:04.866507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 60:
255.255.255.255.31337 > 207.166.253.145.printer: R [bad tcp cksum b5b5!]
0:3(3) ack 0 win 0 [RST cko] (ttl 15, id 0, len 43, bad cksum 28e0!)
0x0000       4500 002b 0000 0000 0f06 28e0 ffff ffff  E..+......(.....
0x0010       cfa6 fd91 7a69 0203 0000 0000 0000 0000  ....zi..........
0x0020       5014 0000 de07 0000 636b 6f00 0000       P.......cko...
```

Description of attack:

The signature of the attack is for the Trojan named “Q” created by Mixter. The “Q” Trojan allows for remote execution of commands. These are typically run as “root”. The Trojan also uses encryption. This attack is referenced by Common Vulnerabilities and Exposures (CVE) CAN-1999-0660. There were no references found at BugTraq, CERT, or ISS XForce.

This detect involved 34 occurrences to 34 unique destination addresses. I also processed raw logs files “2002.10.1” to “2002.10.10” from Incidents.org spanning 10/31/02 to 11/10/02. This resulted in 411 occurrences to 406 unique destinations. All of the detects contained the same packet profile; payload data “cko” or 0x636B6F, TTL of 15, sequence number of 0, ACK and RST flags set, invalid 255.255.255.255 source address, source port of 31337 and destination port of 515. In fact, all packets were identical except the destination IP and checksum values.

This appears to me to be stimulus. This is based strictly on the signature that generated the alert. Since we do not have access to general traffic logs, it is impossible to determine if these packets were a response. It is clear from the logs, that there were no other alerts to the 34 destinations of this detect. This was determined using the Snort log directories. It is also clear that these 34 alerts were the only traffic from a source of 255.255.255.255. This was determined using the command:

```
# tcpdump –r 2002.10.1 src 255.255.255.255 | wc –l
```

This returned only 34 records.

The other aspect I considered was a possible attack on a port 515 or ‘lpr’ vulnerability. However, even if the packet were to reach a target destination with a vulnerable service, the correct response to a lone packet with ACK and RST would be to silently drop the packet. If there had been an active connection, it would be immediately terminated and all associated resources released. However, there could have never been an established session between the target and the 255.255.255.255 address. In addition to this, there were no indications in the log file of any of the common ‘lpd’ attacks such as Ramen, Adore, or lpdw0rm.

If we were to assume that this detect is indeed “Q”, then the “cko” payload is most likely not an encrypted command to run because it is too small. Mike Wyman and Les Gordon both did some testing with “Q” for their GCIA Practicals. However, I believe their tests were flawed because they entered the command of “cko”.

If the detects we saw were encrypted, entering the command of “cko” would not give a payload of “cko”. If it was not encrypted, then what the heck is the command “cko”? I completed a search of the latest ‘Q’ source code and did not find the pattern ‘cko’. However, the tests did prove one thing, the encryption created a very large amount of payload data compared to the command, “cko”. Based on the amount of payload data generated in the Wyman and Gordon tests, it is un-likely that the “cko” payload is an encrypted command, at least not with the default version of “Q”.

Attack mechanisms:

The “Q” Trojan uses raw tcp/udp/icmp packets to send remote commands to a machine running the server or daemon component. The client component is called ‘q’ and the server component is called ‘qd’ and can be used stand alone. There is also a component called ‘qs’ which send commands to be executed remotely or control commands for the server over the tunnel created by ‘q’ and ‘qd’. Using this Trojan, the user can control the remote host. Commands can be executed or attacks can be launched at other targets.

There are a number of other exploits, such as lpdw0rm, Ramen, and Adore WORMs, that take advantage of vulnerabilities of the LPRng service running on port 515. However, I found no indications of these WORMs in the log file. I looked specifically for evidence of Ramen, but found none:

```
# tcpdump –r 2002.10.1 port 27374
```

Based on the way ‘Q’ works, these detects could have been attempts to contact a ‘Q’ server running on port 515 one of the targets. These attempts might have been made to cause infected hosts to “phone home”. However, based on the Wyman and Gordon tests, the traffic in this detect does not follow the same pattern as a version of ‘Q’.

I searched the source code of ‘Q’ 2.4 and did not find any match for “cko”. So, I also searched for the decimal and hex equivalents of ‘cko’ with no luck. “What if this is some type of address?” I asked myself. Maybe ‘cko’ or “636B6F” was the netblock “99.107.111”. However, this is part of an IANA reserved range. Based on all of the evidence, I decided that the traffic must be the result of one of the following three possibilities.

One: The traffic is some sort of modified version of ‘Q’ or the ‘libmix’ library that obviously doesn’t work, or some other tool that has been modified incorrectly. This would make the traffic annoying, but harmless.

Two: The traffic is just crafted traffic designed to trigger ID systems. A kind of “blue herring” designed to occupy IDS Analysts. In this case the traffic is very annoying, but still harmless.

Three: The traffic is some new WORM or Trojan that has not delivered a payload, Yet! This new malware could be lying dormant waiting to strike.

The “non-paranoid” part of me, as small as that is, wants to believe it is scenario one. The cynical part of me believes it is scenario two. However, my gut tells me it is the latter because I just can’t figure out why this traffic is so common across such a wide date range. I know I will continue to watch for this traffic on the networks I monitor.

Correlations:

Additional information and the source code for 'Q' can be found at the author's web site:
http://mixter.warrior2k.com/

Information regarding the Adore worm can be found at:
http://www.iss.net/security_center/static/6681.php

Additional information regarding the Ramen worm can be found at:
http://www.iss.net/security_center/static/6544.php
http://www.cert.org/incident_notes/IN-2001-01.html

Russell Fulton believes it is just someone attempting to trip or set off Intrusion Detection Systems. This detect was more similar to my detect, but does not contain sequence number, flags, or payload information.
http://lists.jammed.com/incidents/2001/04/0062.html

In June of 2002, an analyst calling themselves "fragga" detected a similar type of packet. The destination port was different and the packets looked like real traffic with valid sequence numbers and valid source IP addresses. However, the payload was the same.
http://online.securityfocus.com/archive/75/279535

Crist Clark believes this type of traffic is the result of a broken worm.
http://lists.jammed.com/incidents/2001/07/0023.html

Tu Miem's GCIA practical stating this type of traffic might be a probe from an IRC site.
http://cert.uni-stuttgart.de/archive/intrusions/2003/02/msg00000.html

Mike Wymann did some research into this traffic and Q2.4 for his GCIA practical.
http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00509.html

Les Gordon also did some research into this traffic and Q2.4 for his GCIA Practical.
http://cert.uni-stuttgart.de/archive/intrusions/2002/10/msg00221.html

Evidence of active targeting:

This destination addresses appears completely random except that they are all in the 207.166/16 net block. I believe this is due to the fact that this “207.166” is the network behind the router. Of course, I believe that this 207.166 range is “munged” or obfuscated and not the actual range.

The timing of the traffic is very slow, and has no discernable pattern indicated by the chart below. The traffic spanned almost two full days.

10/31/2002 17:00:00 - 17:59:59	0	
10/31/2002 18:00:00 - 18:59:59	2	
10/31/2002 19:00:00 - 19:59:59	0	
10/31/2002 20:00:00 - 20:59:59	0	
10/31/2002 21:00:00 - 21:59:59	2	
10/31/2002 22:00:00 - 22:59:59	1	
10/31/2002 23:00:00 - 23:59:59	2	
11/1/2002 0:00:00 - 0:59:59	2	
11/1/2002 1:00:00 - 1:59:59	1	
11/1/2002 2:00:00 - 2:59:59	3	
11/1/2002 3:00:00 - 3:59:59	2	
11/1/2002 4:00:00 - 4:59:59	0	
11/1/2002 5:00:00 - 5:59:59	1	
11/1/2002 6:00:00 - 6:59:59	5	
11/1/2002 7:00:00 - 7:59:59	0	
11/1/2002 8:00:00 - 8:59:59	2	
11/1/2002 9:00:00 - 9:59:59	3	
11/1/2002 10:00:00 - 10:59:59	0	
11/1/2002 11:00:00 - 11:59:59	0	
11/1/2002 12:00:00 - 12:59:59	3	
11/1/2002 13:00:00 - 13:59:59	2	
11/1/2002 14:00:00 - 14:59:59	0	
11/1/2002 15:00:00 - 15:59:59	0	
11/1/2002 16:00:00 - 16:59:59	2	
11/1/2002 17:00:00 - 17:59:59	1	
11/1/2002 18:00:00 - 18:59:59	0	

Figure 1

Severity:

(3 + 1) – (3 + 4) = -3

(criticality + lethality) – (system countermeasures + network countermeasures) = severity

Criticality=3, Lethality=1, System counter measures=3, Network counter measures=4
Severity = -3

Since I have no information regarding the function of the target devices, or the system countermeasures in place, I have assigned them a median value of 3. Based on the evidence, even if this packet were allowed to the destination, it would cause no damage. Therefore Lethality was assigned a value of 1. Based on speculation of the network design of an IDS between two Cisco devices, I made the assumption one of the devices was a Firewall, therefore I assigned a value of 4 to network countermeasures.

Defensive recommendations:

I have made the assumption that this site employed a Firewall and an IDS. Therefore, I recommend that the Firewall configuration block all inbound port 515 traffic. This should be the case anyway.

Multiple choice test question:

```
11/01/2002 00:10:04.866507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 60:
255.255.255.255.31337 > 207.166.253.145.printer: R [bad tcp cksum b5b5!]
0:3(3) ack 0 win 0 [RST cko] (ttl 15, id 0, len 43, bad cksum 28e0!)
0x0000       4500 002b 0000 0000 0f06 28e0 ffff ffff  E..+......(.....
0x0010       cfa6 fd91 7a69 0203 0000 0000 0000 0000  ....zi..........
0x0020       5014 0000 de07 0000 636b 6f00 0000       P.......cko...
```

When a machine receives a TCP packet with the RST and ACK flags set, the correct response is:

a. Send an ACK and then close the connection
b. Wait for the time out period and then close the connection
c. Send a FIN and ACK and wait for a reply
d. Close the connection immediately
e. None of the above

Answer: d – close the connection immediately and release all associated buffers.

Detect Submitted:

February 7, 2003
http://cert.uni-stuttgart.de/archive/intrusions/2003/02/msg00066.html

Comments from Incidents.org Posting

http://cert.uni-stuttgart.de/archive/intrusions/2003/02/msg00067.html
From: Andrew Rucker Jones [mailto:arjones@simultan.dyndns.org]
Sent: Saturday, February 08, 2003 11:15 AM
To: Ron Shuck; intrusions@incidents.org
Subject: Re: LOGS: GIAC GCIA Version 3.3 Practical Detect

I think an important point here is that Q generates a key for encryption during compilation (if i recall a previous analysis correctly). That key should be different for every compiled version, meaning if the "cko" is encrypted, there is no way for any of us to analyze what the plaintext might be.

Why does it necessarily have to be a broken piece of malware? Why can't it be innocent traffic from a broken TCP/IP stack or application? You don't know what the environment looks like, and it's entirely possible that someone there is manufacturing new equipment with a locally developed TCP/IP implementation, or that someone has a computer that is 10 or 15 years old. I know this is the case where i work, and we have some strange packets on our network because of it. What do You think? Let me give You a great example. We have printing (spooling) software for a large printer, and a big customer of ours prints on our printer. The printing software sometimes fails to close the TCP connection properly, leading to a perpetually open connection on the spooler. For some stupid reason beyond me, the program that our customer uses starts any new set of print jobs by using the last source port from the last set. Like i say, don't ask me why. Stupid program. Due to the still open connection on the spooler, this doesn't work, and the spooler has to be restarted every time. We briefly considered using hping to send a reset packet to kill the connection. If we had done that in an automated way, it might well have looked somewhat similar to this traffic (though with the correct source address). See what I'm saying? Just a thought.

Response:
Andrew is right, there's no way to analyze what the plain text might be, but I think it's safe to assume that if encryption is taking place, the payload would not be 'cko' for a command of 'cko'. Otherwise, it's not encrypted.

I though about whether the traffic was malware or just broken code as well. Andrew's example helps, but I still stick with malware. Here's my reason. If this is just broken innocuous code, why is it communicating with as many different hosts as indicated by my detect and the other detects recently? I suggest that if you have broken code, on the Internet, blasting packets outside of your network, it is malware whether it is intended that way or not. I didn't see this traffic at any of the sites I watch, but if I did I would have spent time looking into it. So, by my definition it is malware because it causes a negative impact on my network even if that was not the intention.

I totally agree that I should make that distinction in the analysis. The traffic may not be intentional broken malware.

From: Gordon, Les M [Les.M.Gordon@team.telstra.com]
Sent: Sunday, February 09, 2003 6:01 PM
To: Ron Shuck
Subject: RE: LOGS: GIAC GCIA Version 3.3 Practical Detect

Ron,

Just one quick point is that versions of Q prior to 2.0 did NOT encrypt the payload used for requesting the server to execute commands. My analysis covered versions prior to 2.0 as well as 2.4.

Regards,
Les

Response:
Good clarification. Versions of Q prior to 2.0 did not encrypt the payload. So, if my detect was utilizing a version prior to 2.0, the then 'cko' payload would have been plain text. In that case, the test is valid and demonstrates that the version 1.0 did not create an identical packet as our detect packets.

If the detect was utilizing a version later than 2.0, testing with the command of 'cko' is still flawed. I stand by my original statement. If the payload was assumed encrypted, testing with a command of 'cko' could not produce the packets of the detect. If the payload was indeed encrypted the 'cko' payload would have to be different than the command, or it wouldn't be very good encryption.

So, the clarification is that the first part of the testing was valid and provided good information, but the second test was not valid. However, it did provide some useful information. It indicated that a command of 3 characters created a payload of 86 bytes. It seems highly unlikely that any command would create an encrypted payload of just 3 bytes. This rules out the possibility, in my mind, that the 'cko' payload of the detect was an encrypted payload for some unknown command.

http://cert.uni-stuttgart.de/archive/intrusions/2003/02/msg00204.html
From: Robert Wagner [rwagner@eruces.com]
Sent: Monday, February 24, 2003 10:41 AM
To: Ron Shuck; intrusions@incidents.org
Subject: RE: LOGS: GIAC GCIA Version 3.3 Practical Detect

First one - Backdoor Q Access.

10/31-18:10:04.866507 255.255.255.255:31337 -> 207.166.253.145:515 "Defensive recommendations: I have made the assumption that this site employed a Firewall and an IDS. Therefore, I recommend that the Firewall configuration block all inbound port 515 traffic. This should be the case anyway."

I just saw the packet and read your defense, you may have covered this somewhere in the documentation. I didn't see the mention of ingress and egress filters. Do you think they are important?

Response:
This is an excellent point that I did not mention in my defensive recommendations. Ingress and egress filters can provide an additional layer of protection. As we all know, the key to security is layers. For this particular detect, an ingress filter may have proved the most effective

Detect 2: BAD TRAFFIC IP reserved bit set

This output is the standard log output from Snort, and has the following format;

```
Alert Signature Name
Date - Time  Source IP:Source Port  -> Destination IP:Destination Port
IP Header Information
Protocol Information

[**] BAD TRAFFIC ip reserved bit set [**]
11/06-18:52:03.576507 200.200.200.1 -> 207.166.81.197
TCP TTL:242 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset: 0x0864   Frag Size: 0xFFFFF7B0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

 [**] BAD TRAFFIC ip reserved bit set [**]
11/06-21:09:28.956507 200.200.200.1 -> 207.166.119.251
TCP TTL:242 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset: 0x0864   Frag Size: 0xFFFFF7B0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

 [**] BAD TRAFFIC ip reserved bit set [**]
11/07-00:05:36.006507 200.200.200.1 -> 207.166.15.33
TCP TTL:242 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset: 0x0864   Frag Size: 0xFFFFF7B0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

 [**] BAD TRAFFIC ip reserved bit set [**]
11/07-12:02:09.736507 200.200.200.1 -> 207.166.219.11
TCP TTL:242 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset: 0x0864   Frag Size: 0xFFFFF7B0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

 [**] BAD TRAFFIC ip reserved bit set [**]
11/07-15:38:43.136507 200.200.200.1 -> 207.166.14.185
TCP TTL:242 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset: 0x0864   Frag Size: 0xFFFFF7B0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

 [**] BAD TRAFFIC ip reserved bit set [**]
11/07-15:43:44.736507 200.200.200.1 -> 207.166.37.247
TCP TTL:242 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset: 0x0864   Frag Size: 0xFFFFF7B0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```


Source of Trace:

The source of the trace was the raw logs directory at Incidents.org. The log files are the result of a Snort instance running in binary logging mode. The logs themselves have been sanitized. The specific file used for this detect was;
http://www.incidents.org/logs/Raw/2002.10.7

Since the network layout was not provided, I can only assume, but there is strong evidence that the Snort sensor that captured the data was located between two Cisco devices. Not only the packets involved in this detect, but all packets from the above log file have one of two distinct MAC addresses. There are two different OUIs and both belong to Cisco. My guess would be that the two devices are an external router and a PIX firewall or NAT router.

```
11/07/2002 00:52:03.576507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
200.200.200.1 > 207.166.81.197: (frag 0:20@17184)
11/07/2002 03:09:28.956507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
200.200.200.1 > 207.166.119.251: (frag 0:20@17184)
11/07/2002 06:05:36.006507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
200.200.200.1 > 207.166.15.33: (frag 0:20@17184)
11/07/2002 18:02:09.736507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
200.200.200.1 > 207.166.219.11: (frag 0:20@17184)
11/07/2002 21:38:43.136507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
200.200.200.1 > 207.166.14.185: (frag 0:20@17184)
11/07/2002 21:43:44.736507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 60:
200.200.200.1 > 207.166.37.247: (frag 0:20@17184)
```

The other evidence is that all of the packets of this detect, and all of the packets in the log file, contain a bad IP header checksum. This was completed visually in Ethereal for the packets of the detect, and using the following commands for the entire log file.

```
# tcpdump –vnr 2002.10.7 | wc –l
# tcpdump –vnr 2002.10.7 | grep “bad cksum” | wc –l
```

Both commands returned the same result, 2509 records. This is evidence that the protected network addresses were “munged” or obfuscated as stated in the README file in the log directory at Incidents.org. I also did a sorted visual scan of both source and destination addresses of the entire log file. I did not detect any private networks. This is in no way conclusive, but it leads me to believe that the sensor was located outside the firewall or NAT device, and inside an external router providing connectivity to the Internet.

Detect was generated by:

The detect was generated using Snort v 1.9.0 (Build 209) with a default “snort.conf” v1.110 and “bad-traffic.rules” v1.18 on Red Hat Linux 7.3. The detect was analyzed using Snort log files, ACID, Ethereal, and Tcpdump.

The specific rule that triggered the alert was:

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"BAD TRAFFIC ip reserved bit
set"; fragbits:R; sid:523;  classtype:misc-activity; rev:3;)
```

The rule basically alerts on any traffic with the Reserved Bit set in the IP header. The Reserved Bit is the high order bit of the 7th byte of the IP header (byte 6). This bit is not used and should be set to 0.

Probability the source address was spoofed:

Although sufficient data is not available to determine with any certainty, there is no real evidence that the source address was spoofed. It should be noted that the packets do exhibit signs of packet crafting. Based on the information available, I would estimate the probability at 50% or .5. The 200.128/9 net block is registered to Comite Gestor da Internet no Brasil. The 200.200/16 Class B is registered to Embratel-Empresa Brasileira de Telecomunicações SA. This is a Phone company in Brazil that provides Internet services.

Description of attack:

This detect consisted of 6 events (displayed below in tcpdump format). They all had the same source 200.200.200.1 and 6 different targets. I also searched the log files "2002.10.1" through "2002.10.14" at Incidents.org, and found 9 more events from the same source and 9 different targets. For the "2002.10.7" log file, I found that there was no other traffic that triggered an alert involving the destinations of this detect.

```
11/07/2002 00:52:03.576507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 60: 200.200.200.1
> 207.166.81.197: (frag 0:20@17184) (ttl 242, len 40, bad cksum d87f!)
0x0000       4500 0028 0000 8864 f206 d87f c8c8 c801  E..(...d........
0x0010       cfa6 51c5 1022 0050 3b99 8046 3b99 8046  ..Q.".P;..F;..F
0x0020       0004 0000 0fc4 0000 0000 0000 0000       ..............
11/07/2002 03:09:28.956507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 60: 200.200.200.1
> 207.166.119.251: (frag 0:20@17184) (ttl 242, len 40, bad cksum b249!)
0x0000       4500 0028 0000 8864 f206 b249 c8c8 c801  E..(...d...I....
0x0010       cfa6 77fb 0fdd 0050 3c17 51d8 3c17 51d8  ..w....P<.Q.<.Q.
0x0020       0004 0000 45b3 0000 0000 0000 0000       ....E.........
11/07/2002 06:05:36.006507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 60: 200.200.200.1
> 207.166.15.33: (frag 0:20@17184) (ttl 242, len 40, bad cksum 1c25!)
0x0000       4500 0028 0000 8864 f206 1c25 c8c8 c801  E..(...d...%....
0x0010       cfa6 0f21 0fa6 0050 3cb8 90a2 3cb8 90a2  ...!...P<...<...
0x0020       0004 0000 30ef 0000 0000 0000 0000       ....0.........
11/07/2002 18:02:09.736507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 60: 200.200.200.1
> 207.166.219.11: (frag 0:20@17184) (ttl 242, len 40, bad cksum 4e3b!)
0x0000       4500 0028 0000 8864 f206 4e3b c8c8 c801  E..(...d..N;....
0x0010       cfa6 db0b 1354 0050 3f48 9dc6 3f48 9dc6  .....T.P?H..?H..
0x0020       9104 0000 aeee 0000 0000 0000 0000       ..............
11/07/2002 21:38:43.136507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 60: 200.200.200.1
> 207.166.14.185: (frag 0:20@17184) (ttl 242, len 40, bad cksum 1d8b!)
0x0000       4500 0028 0000 8864 f206 1d8b c8c8 c801  E..(...d........
0x0010       cfa6 0eb9 0dc7 0050 400e e378 400e e378  .......P@..x@..x
0x0020       9104 0000 f6da 0000 0000 0000 0000       ..............
11/07/2002 21:43:44.736507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 60: 200.200.200.1
> 207.166.37.247: (frag 0:20@17184) (ttl 242, len 40, bad cksum 44e!)
0x0000       4500 0028 0000 8864 f206 044e c8c8 c801  E..(...d...N....
0x0010       cfa6 25f7 0d44 0050 4013 7dac 4013 7dac  ..%..D.P@.}.@.}.
0x0020       0004 0000 3ab0 0000 0000 0000 0000       ....:.........
```

I also did some research at DShield.org on the 200.200.200.1 address. There were 1121 records involving this IP along with 443 different targets. However, based on the "whois" information, I believe that the IP is part of a dial-up ISP range.

There is no hard evidence to support this conclusion, but it is my opinion that this traffic is stimulus. By this I mean that the traffic was not the result of a stimulus or request from the protected network. Here is why. First, the source was a single IP that does not resolve to a name. It is possible that it did resolve to a name at the time of the detect, but there is strong evidence that the source IP was and is part of a dial-up ISP. Second, the destinations involved 15 different IP addresses in what appears to be different subnets. Finally, the source was used in the past to target other systems as indicated by the DShield.org data.

Attack mechanisms:

The mechanism of this detect is not clear. There is not an obvious payload or exploit associated with this detect. There are several components of this detect that do warrant further investigation.

First, all of the packets in question have the high order bit of byte 6 set in the IP header. This bit position is reserved and should always have a value of zero, according to RFC791 or STD005. A value in this position is often associated with packet crafting. Typically, this would be used as a fingerprinting mechanism. Unfortunately, the "munging" or obfuscation of the data has eliminated the possibility of determining if the reserved bit was set or was the result of packet corruption. If the value was an error, the checksum for the header would be incorrect. This is not guaranteed, but generally the case. However, all of the packets have incorrect checksums as a result of the obfuscation. I did not find any direct references to this type of packet on CVE. The closest was CAN-1999-0240, but that was related to SYN packets.

Second, all of the datagrams have the same fragment ID of 0. This is highly unlikely under normal circumstances. The value 0 would be a valid value, but the fragment ID should vary. Typically IP stacks increment this number for each datagram. It is possible that the packets that triggered this detect were ID zero, but very improbable that the two that occur in the same hour time frame would have the same ID of zero.

Third, all of the packets in question appear to be the last fragment of a datagram. Again, it seems too coincidental that only the last fragment of 6 different datagrams destined for 6 different addresses would be corrupted in the same fashion affecting the reserved bit of the IP header. It is very common to use fragments as a way to circumvent security controls or avoid detection.

Fourth, all of the packets claim to have a fragment offset of 17184 and contain 20 bytes of data and 6 bytes of trailer. This too seems very coincidental. If these packets were actually the last fragment at offset 17184 for fragment ID 0, then I would contend that the original datagram was the same size in all six instances. However, the data portion is different. I looked for indications of bit flipping or data corruption in the data, but I found nothing that would indicate data corruption.

Finally, all of the packets have a TTL of 242. If this is indeed a default value, it is indicative of a Solaris 2.x machine in close proximity to the destinations. This leads to the real question. What is this detect? I suggest that it is one of the following possibilities:

One: Traffic generated by a broken IP stack on an a network not monitored by an IDS.

Two: Some type of attempted reconnaissance designed to circumvent a Firewall or perimeter security device. The use of the reserved bit could have been designed to provide some type of operating system fingerprinting.

There is only one factor permitting me from labeling this detect as bad traffic from poorly written software. That factor is the number of targets. Bret Wrisley and Soren Macbeth found the same type of traffic during their GCIA practicals.

Correlations:

Information on default TTL values for various operating systems.
http://secfr.nerim.net/docs/fingerprint/en/ttl_default.html

Information at DShield.org regarding the source of this detect.
http://www.dshield.org/ipinfo.php?ip=200.200.200.1

Brent Wrisley submitted a similar detect for his GCIA Practical.
http://cert.uni-stuttgart.de/archive/intrusions/2002/10/msg00079.html

Soren MacBeth submitted a similar detect for his GCIA Practical.
http://cert.uni-stuttgart.de/archive/intrusions/2002/10/msg00119.html

IP reserved bit Common Vulnerability and Exposure candidate.
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0240

Network Working Group Standard for IP. STD005 or RFC791
ftp://ftp.rfc-editor.org/in-notes/std/std5.txt

Information on default TTL values for various operating systems.
http://secfr.nerim.net/docs/fingerprint/en/ttl_default.html

Evidence of active targeting:

There are insufficient numbers of packets to correctly identify active targeting, in my opinion. However, based on the packet craft evidence, the source location, and the type of attack this detect would indicate, I suggest that the traffic was directed at the target addresses. I do not believe that the attack is specifically targeting the protected network. In other words, I believe the evidence indicates that the packets are an attempt at reconnaissance, but not directed specifically at the protected network as an entity.

This is supported by the following evidence. The packets involved only targeted 15 IP addresses over a 15 day period. It could be “low and slow”, but the traffic is too obvious to be “low”. My point is why scan very slowly while waiving a giant red flag.

11/6/2002 17:00:00 - 17:59:59	0	
11/6/2002 18:00:00 - 18:59:59	1	
11/6/2002 19:00:00 - 19:59:59	0	
11/6/2002 20:00:00 - 20:59:59	0	
11/6/2002 21:00:00 - 21:59:59	1	
11/6/2002 22:00:00 - 22:59:59	0	
11/6/2002 23:00:00 - 23:59:59	0	
11/7/2002 0:00:00 - 0:59:59	1	
11/7/2002 1:00:00 - 1:59:59	0	
11/7/2002 2:00:00 - 2:59:59	0	
11/7/2002 3:00:00 - 3:59:59	0	
11/7/2002 4:00:00 - 4:59:59	0	
11/7/2002 5:00:00 - 5:59:59	0	
11/7/2002 6:00:00 - 6:59:59	0	
11/7/2002 7:00:00 - 7:59:59	0	
11/7/2002 8:00:00 - 8:59:59	0	
11/7/2002 9:00:00 - 9:59:59	0	
11/7/2002 10:00:00 - 10:59:59	0	
11/7/2002 11:00:00 - 11:59:59	0	
11/7/2002 12:00:00 - 12:59:59	1	
11/7/2002 13:00:00 - 13:59:59	0	
11/7/2002 14:00:00 - 14:59:59	0	
11/7/2002 15:00:00 - 15:59:59	2	
11/7/2002 16:00:00 - 16:59:59	0	

Figure 2

Severity:

(3 + 1) – (3 + 4) = -3

(criticality + lethality) – (system countermeasures + network countermeasures)
= severity

Criticality=3, Lethality=1, System counter measures=3, Network counter measures=4
Severity = -3

Since I have no information regarding the function of the target devices, or the system countermeasures in place, I have assigned them a median value of 3. Based on the evidence, even if this packet were allowed to the destination, it would cause no damage. Therefore Lethality was assigned a value of 1. Based on speculation of the network design of an IDS between two Cisco devices, I made the assumption one of the devices was a Firewall, therefore I assigned a value of 4 to network countermeasures.

Defensive recommendations:

This type of reconnaissance, if it truly was reconnaissance, should be easily defeated with current versions of Firewalls. I would also watch this specific alert more closely.

Multiple choice test question:

```
11/07/2002 00:52:03.576507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 60: 200.200.200.1
> 207.166.81.197: (frag 0:20@17184) (ttl 242, len 40, bad cksum d87f!)
0x0000       4500 0028 0000 8864 f206 d87f c8c8 c801  E..(...d........
0x0010       cfa6 51c5 1022 0050 3b99 8046 3b99 8046  ..Q.."P;..F;..F
0x0020       0004 0000 0fc4 0000 0000 0000 0000       ..............
```

The fragment ID of zero in the packet above is derived from the following.

A. The IP Identification Number field
B. The Initial Sequence Number (ISN) of the datagram
C. An IP options field
D. A TCP options field
E. None of the above

Answer: A The 16 bit IP Identification Number field of the IP Header

Detect Submitted:

February 10, 2003
http://cert.uni-stuttgart.de/archive/intrusions/2003/02/msg00088.html

Detect 3: P2P Outbound GNUTella client request

This output is the standard log output from Snort, and has the following format;

```
Alert Signature Name
Date - Time  Source IP:Source Port  -> Destination IP:Destination Port
IP Header Information
Protocol Information

[**] P2P Outbound GNUTella client request [**]
11/14-13:46:22.526507 170.129.50.120:61744 -> 142.217.196.48:8330
TCP TTL:123 TOS:0x0 ID:61506 IpLen:20 DgmLen:158 DF
***AP*** Seq: 0x2688C47B  Ack: 0xFE5553  Win: 0x4038  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] P2P Outbound GNUTella client request [**]
11/14-13:46:25.506507 170.129.50.120:61744 -> 142.217.196.48:8330
TCP TTL:123 TOS:0x0 ID:61792 IpLen:20 DgmLen:158 DF
***AP*** Seq: 0x2688C47B  Ack: 0xFE5553  Win: 0x4038  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] P2P Outbound GNUTella client request [**]
11/14-13:46:31.526507 170.129.50.120:61744 -> 142.217.196.48:8330
TCP TTL:123 TOS:0x0 ID:62383 IpLen:20 DgmLen:158 DF
***AP*** Seq: 0x2688C47B  Ack: 0xFE5553  Win: 0x4038  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] P2P Outbound GNUTella client request [**]
11/14-13:46:41.186507 170.129.50.120:61784 -> 142.217.196.48:8330
TCP TTL:123 TOS:0x0 ID:63239 IpLen:20 DgmLen:62 DF
***AP*** Seq: 0x2803691C  Ack: 0xFE9E3B  Win: 0x4038  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] P2P Outbound GNUTella client request [**]
11/14-13:46:43.616507 170.129.50.120:61744 -> 142.217.196.48:8330
TCP TTL:123 TOS:0x0 ID:63458 IpLen:20 DgmLen:158 DF
***AP**F Seq: 0x2688C47B  Ack: 0xFE5553  Win: 0x4038  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] P2P Outbound GNUTella client request [**]
11/14-13:46:47.176507 170.129.50.120:61784 -> 142.217.196.48:8330
TCP TTL:123 TOS:0x0 ID:63787 IpLen:20 DgmLen:62 DF
***AP*** Seq: 0x2803691C  Ack: 0xFE9E3B  Win: 0x4038  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```


Source of Trace:

The source of the trace was the raw logs directory at Incidents.org. The log files are the result of a Snort instance running in binary logging mode. The logs themselves have been sanitized. The specific file used for this detect was;
http://www.incidents.org/logs/Raw/2002.10.14

Since the network layout was not provided, I can only assume, but there is strong evidence that the Snort sensor that captured the data was located between two Cisco devices. Not only the packets involved in this detect, but all packets from the above log file have one of two distinct MAC addresses. There are two different OUIs and both belong to Cisco. Based on this, my guess would be that the two devices are an external router and a PIX firewall or NAT router. This was my guess for my previous 2 detects for this practical. However, in the previous two detects one of the IP addresses appeared to be obfuscated.

Detect 1 example:

```
11/01/2002 00:10:04.866507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 60:
255.255.255.255.31337 > 207.166.253.145.printer: R [bad tcp cksum b5b5!]
0:3(3) ack 0 win 0 [RST cko] (ttl 15, id 0, len 43, bad cksum 28e0!)
0x0000       4500 002b 0000 0000 0f06 28e0 ffff ffff  E..+......(.....
0x0010       cfa6 fd91 7a69 0203 0000 0000 0000 0000  ....zi..........
0x0020       5014 0000 de07 0000 636b 6f00 0000       P.......cko...
```

Detect 2 example:

```
11/07/2002 00:52:03.576507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 60: 200.200.200.1
> 207.166.81.197: (frag 0:20@17184) (ttl 242, len 40, bad cksum d87f!)
0x0000       4500 0028 0000 8864 f206 d87f c8c8 c801  E..(...d........
0x0010       cfa6 51c5 1022 0050 3b99 8046 3b99 8046  ..Q.."P;..F;..F
0x0020       0004 0000 0fc4 0000 0000 0000 0000       ..............
```

The bad checksums corroborated this conclusion. This detect has the same two Cisco devices for all traffic, but does not have the bad checksum issue. The creation date on the log files are different between this detect, and my previous detects. This means that the obfuscation would be different, but does not explain the lack of bad check sums.

All of this leads me to believe that this is the same network from my previous two detects, and that the obfuscation of this log file corrected the checksums. I would also deduce that the 170.129.50.120 is the obfuscated address. This is based on the traffic direction of the previous two detects. The `0:0:c:4:b2:33` MAC always appeared to be the inside router or PIX (destination). If that is the case, then the traffic in this detect is outbound since the `0:0:c:4:b2:33` MAC is the source.

```
11/14/2002 19:46:22.526507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 172:
170.129.50.120.61744 > 142.217.196.48.8330: tcp 118 (DF)
11/14/2002 19:46:25.506507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 172:
170.129.50.120.61744 > 142.217.196.48.8330: tcp 118 (DF)
11/14/2002 19:46:31.526507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 172:
170.129.50.120.61744 > 142.217.196.48.8330: tcp 118 (DF)
11/14/2002 19:46:41.186507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 76:
170.129.50.120.61784 > 142.217.196.48.8330: tcp 22 (DF)
11/14/2002 19:46:43.616507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 172:
170.129.50.120.61744 > 142.217.196.48.8330: tcp 118 (DF)
11/14/2002 19:46:47.176507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 76:
170.129.50.120.61784 > 142.217.196.48.8330: tcp 22 (DF)
```

Detect was generated by:

The detect was generated using Snort v 1.9.0 (Build 209) with a default "snort.conf" v1.110 and "p2p.rules" v1.8 on Red Hat Linux 7.3. The detect was analyzed using Snort log files, ACID, Ethereal, and Tcpdump.

The specific rule that triggered the alert was:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"P2P Outbound GNUTella
client request"; flow:to_server,established; content:"GNUTELLA CONNECT";
depth:40; classtype:misc-activity; sid:556;  rev:4;)
```

This alert triggers on any source or destination address or port when the following conditions are met. First, the packet is part of an established TCP session. This is caused by the "flow:established" option. Basically this does not inspect packets with SYN. Second, the packet must be from client to server. This is caused by the "flow:to_server" option. Basically, this means traffic from the machine that initiated the session (client) to the server. Third, the payload of the packet must have the text "GNUTELLA CONNECT" in the first 40 bytes.

Probability the source address was spoofed:

I would estimate the probability that these packets had spoofed source addresses at .1 or 10%. There is no real evidence of packet crafting, and it is my position that these packets are outbound traffic. It would make no sense to for a client request to spoof the source. Although I do not believe these packets had a spoofed source address, there is strong evidence that the source address is obfuscated.

Description of attack:

The first three packets appear to be the same session. The first was the original, and the next two are retries. This is evident in the same sequence number in all three packets, and the fact that they are spaced 3, 6, 9 seconds apart. The fifth packet is sent 12 seconds later, and is a request to terminate the connection with the FIN flag set.

The fourth and sixth packets are related. They both have the same sequence number and are spaced 3 seconds apart. These packets contain a different payload than packets 1,2,3, and 5.

```
11/14/2002 19:46:22.526507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 172:
midgaard.smsc.com.61744 > sehv-3-lt-48.lino.sympatico.ca.8330: P [tcp sum ok]
646497403:646497521(118) ack 16667987 win 16440 (DF) (ttl 123, id 61506, len 158)
0x0000       4500 009e f042 4000 7b06 df13 aa81 3278  E....B@.{.....2x
0x0010       8ed9 c430 f130 208a 2688 c47b 00fe 5553  ...0.0..&..{..US
0x0020       5018 4038 7eea 0000 474e 5554 454c 4c41  P.@8~...GNUTELLA
0x0030       2043 4f4e 4e45 4354 2f30 2e36 0d0a 5573  .CONNECT/0.6..Us
0x0040       6572 2d41 6765 6e74 3a20 4d6f 7270 6865  er-Agent:.Morphe
0x0050       7573 2032 2e30 2e30 2e38 0d0a 582d 556c  us.2.0.0.8..X-Ul
0x0060       7472 6170 6565 723a 2046 616c 7365 0d0a  trapeer:.False..
0x0070       5245 5155 4553 5454 4553 5443 4f4e 4e3a  REQUESTTESTCONN:
0x0080       2039 3531 320d 0a4c 6973 7465 6e69 6e67  .9512..Listening
0x0090       506f 7274 203a 3935 3132 0d0a 0d0a       Port.:9512....
```

```
11/14/2002 19:46:25.506507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 172:
midgaard.smsc.com.61744 > sehv-3-lt-48.lino.sympatico.ca.8330: P [tcp sum ok]
0:118(118) ack 1 win 16440 (DF) (ttl 123, id 61792, len 158)
0x0000       4500 009e f160 4000 7b06 ddf5 aa81 3278  E....`@.{.....2x
0x0010       8ed9 c430 f130 208a 2688 c47b 00fe 5553  ...0.0..&..{..US
0x0020       5018 4038 7eea 0000 474e 5554 454c 4c41  P.@8~...GNUTELLA
0x0030       2043 4f4e 4e45 4354 2f30 2e36 0d0a 5573  .CONNECT/0.6..Us
0x0040       6572 2d41 6765 6e74 3a20 4d6f 7270 6865  er-Agent:.Morphe
0x0050       7573 2032 2e30 2e30 2e38 0d0a 582d 556c  us.2.0.0.8..X-Ul
0x0060       7472 6170 6565 723a 2046 616c 7365 0d0a  trapeer:.False..
0x0070       5245 5155 4553 5454 4553 5443 4f4e 4e3a  REQUESTTESTCONN:
0x0080       2039 3531 320d 0a4c 6973 7465 6e69 6e67  .9512..Listening
0x0090       506f 7274 203a 3935 3132 0d0a 0d0a       Port.:9512....
11/14/2002 19:46:31.526507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 172:
midgaard.smsc.com.61744 > sehv-3-lt-48.lino.sympatico.ca.8330: P [tcp sum ok]
0:118(118) ack 1 win 16440 (DF) (ttl 123, id 62383, len 158)
0x0000       4500 009e f3af 4000 7b06 dba6 aa81 3278  E.....@.{.....2x
0x0010       8ed9 c430 f130 208a 2688 c47b 00fe 5553  ...0.0..&..{..US
0x0020       5018 4038 7eea 0000 474e 5554 454c 4c41  P.@8~...GNUTELLA
0x0030       2043 4f4e 4e45 4354 2f30 2e36 0d0a 5573  .CONNECT/0.6..Us
0x0040       6572 2d41 6765 6e74 3a20 4d6f 7270 6865  er-Agent:.Morphe
0x0050       7573 2032 2e30 2e30 2e38 0d0a 582d 556c  us.2.0.0.8..X-Ul
0x0060       7472 6170 6565 723a 2046 616c 7365 0d0a  trapeer:.False..
0x0070       5245 5155 4553 5454 4553 5443 4f4e 4e3a  REQUESTTESTCONN:
0x0080       2039 3531 320d 0a4c 6973 7465 6e69 6e67  .9512..Listening
0x0090       506f 7274 203a 3935 3132 0d0a 0d0a       Port.:9512....
11/14/2002 19:46:41.186507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 76:
midgaard.smsc.com.61784 > sehv-3-lt-48.lino.sympatico.ca.8330: P [tcp sum ok]
671312156:671312178(22) ack 16686651 win 16440 (DF) (ttl 123, id 63239, len 62)
0x0000       4500 003e f707 4000 7b06 d8ae aa81 3278  E..>..@.{.....2x
0x0010       8ed9 c430 f158 208a 2803 691c 00fe 9e3b  ...0.X..(.i....;
0x0020       5018 4038 6675 0000 474e 5554 454c 4c41  P.@8fu..GNUTELLA
0x0030       2043 4f4e 4e45 4354 2f30 2e34 0a0a       .CONNECT/0.4..
11/14/2002 19:46:43.616507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 172:
midgaard.smsc.com.61744 > sehv-3-lt-48.lino.sympatico.ca.8330: FP [tcp sum
ok] 0:118(118) ack 1 win 16440 (DF) (ttl 123, id 63458, len 158)
0x0000       4500 009e f7e2 4000 7b06 d773 aa81 3278  E.....@.{..s..2x
0x0010       8ed9 c430 f130 208a 2688 c47b 00fe 5553  ...0.0..&..{..US
0x0020       5019 4038 7ee9 0000 474e 5554 454c 4c41  P.@8~...GNUTELLA
0x0030       2043 4f4e 4e45 4354 2f30 2e36 0d0a 5573  .CONNECT/0.6..Us
0x0040       6572 2d41 6765 6e74 3a20 4d6f 7270 6865  er-Agent:.Morphe
0x0050       7573 2032 2e30 2e30 2e38 0d0a 582d 556c  us.2.0.0.8..X-Ul
0x0060       7472 6170 6565 723a 2046 616c 7365 0d0a  trapeer:.False..
0x0070       5245 5155 4553 5454 4553 5443 4f4e 4e3a  REQUESTTESTCONN:
0x0080       2039 3531 320d 0a4c 6973 7465 6e69 6e67  .9512..Listening
0x0090       506f 7274 203a 3935 3132 0d0a 0d0a       Port.:9512....
11/14/2002 19:46:47.176507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 76:
midgaard.smsc.com.61784 > sehv-3-lt-48.lino.sympatico.ca.8330: P [tcp sum ok]
0:22(22) ack 1 win 16440 (DF) (ttl 123, id 63787, len 62)
0x0000       4500 003e f92b 4000 7b06 d68a aa81 3278  E..>.+@.{.....2x
0x0010       8ed9 c430 f158 208a 2803 691c 00fe 9e3b  ...0.X..(.i....;
0x0020       5018 4038 6675 0000 474e 5554 454c 4c41  P.@8fu..GNUTELLA
0x0030       2043 4f4e 4e45 4354 2f30 2e34 0a0a       .CONNECT/0.4..
```

So, there are two separate sessions involved. The first is sequence number 646497403 which is also an ACK for 16667987. This appears to be a GNUTella attempt for version 0.6. The second is sequence number 671312156 which is an ACK for 16686651. This appears to be a GNUTella attempt for version 0.4.

This seems to indicate that two connection attempts were made, most likely, from a Window NT or 2000 machine based on the TTL values and window size. It is also evident that the initial three way handshake was completed. However, we did not capture a "GNUTELLA OK" message. This suggests one of two possibilities. First, the target or destination machine was listening on port 8330, but not for the GNUTella protocol. Second, the target machine was running GNUTella, but was no accepting connections for some reason.

Attack mechanisms:

GNUTella is a peer to peer (P2P) based file sharing client. The original version was written by Justin Frankel and Tom Pepper, but many different versions of clients are now available. GNUTella software acts as both a client and a server earning the name "servent".

Basically, a "servent" opens a TCP connection to another "servent". It then sends the "GNUTELLA CONNECT" message along with the version of the client software. The "servent" acting as a server would then respond with "GNUTELLA OK" to accept the connection. In some versions, the "OK" message also contains the version of the software. Also during this process, the new "servent" provides the IP and port that it is listening on for connections. Once a connection is established, the "servent" can search other "servents" for files and provide files.

The software utilizes a set of protocol descriptors to communicate with other "servents". The "ping" and "pong" descriptors are used to identify other "servents" on the network. The "query", and "query hit" descriptors are used to locate files to download, and the "push" descriptor allows for circumventing Firewalls, etc. The "push" descriptor provides the ability of a "servent" that contained a requested file, to establish a connection to the requestor. GNUTella software also includes a type of routing capability as well. "Servents" are expected to route GNUTella traffic based on a set of rules or guidelines for the protocol.

There are several security issues with GNUTella software. First, it may allow the sharing of files in a manner outside of the approved mechanism for a company. There may be mechanisms in place that protect corporate data via the approved methods, but may not be in place for GNUTella traffic. The fact that GNUTella can run over ports that may be allowed by the Firewall such as SMTP or HTTP, is of particular cause for alarm. Second, the use of GNUTella may also bypass a layer of Anti-virus protection. Third, often times these GNUTella clients have spyware or other Trojans included.

Correlations:

The target was not listed at DShield.org, and the Neohapsis Port list does not contain any reference to port 8330. This supports the hypothesis that the target was not providing GNUTella, at least not on port 8330, but it does not provide any additional information to the service that was listening on port 8330. A subsequent search of Google also failed to shed any light on what service would be running on port 8330.
http://www.dshield.org/ipinfo.php?ip=142.217.196.48&Submit=Submit
http://www.neohapsis.com/neolabs/neo-ports/neo-ports.html

Information on Passive OS Fingerprinting used to speculate source system.
http://project.honeynet.org/papers/finger/traces.txt

Information on GNUTella
http://www.sans.org/rr/threats/gnutella.php

These sites contained the information on GNUTella and its protocol that allowed a positive identification of the traffic payload as GNUTella traffic.
http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf
http://rfc-gnutella.sourceforge.net/Proposals/Handshake_06/Gnutella06.txt

Information regarding Spyware and threats of P2P software.
http://www.unwantedlinks.com/Guntella-alert.htm

I only found one comparable detect of GNUTella submitted for a GCIA practical, and that was the one submitted by Nils Reichen.
http://cert.uni-stuttgart.de/archive/intrusions/2002/10/msg00072.html

Evidence of active targeting:

There was no evidence that the protected site was targeted. In fact, there is strong evidence that the detect was generated from an internal system attempting to connect to an external GNUTella “servent”.

Severity:

(2 + 4) – (3 + 1) = 2

(criticality + lethality) – (system countermeasures + network countermeasures)
= severity

Criticality=2, Lethality=4, System counter measures=3, Network counter measures=1

Since I have no information regarding the function of the target devices, or the system countermeasures in place, I would typically assign them a median value of 3. However, based on the TTL information and the GNUTella usage, I would suggest that the source is probably a workstation. Therefore I lowered the criticality to 2. Based on the evidence, I believe this traffic was allowed to the destination. Since it poses a significant risk to the organization, lethality was assigned a value of 4.

Based on speculation of the network design of an IDS between two Cisco devices, I made the assumption one of the devices was a Firewall. However I feel the network countermeasure were ineffective in this case. Therefore, I assigned a value of 1 to network countermeasures.

Defensive recommendations:

My primary recommendation is to implement a company policy that prohibits use of the P2P clients. It could be added to the Acceptable Use policy or a new policy. Make sure the policy is well communicated to the users. Next, tighten Firewall security. Only allow outbound traffic on ports that are required. Do not just allow all outbound connections.

Also, consider use of a proxy server for common outbound services. This could be a simple proxy server used to authenticate connections, or a proxy server based firewall used to inspect the traffic. A proxy server based firewall would be used to stop traffic that was not consistent with the protocol. The proxy server based firewall understands the application layer and only allows traffic that is valid for that application. A simple proxy server or SOCKS server can provide authentication and logging of traffic. Although, this type of proxy would not stop GNUTella traffic, it could require the user to authenticate and then log the traffic. The downside to both types of proxy servers is that the applications on the client machines must support the use of a proxy.

Finally, consider use of active response on IDS for GNUTella detects. Many systems can send TCP RST packets based on a signature match. This should be considered for the outbound “servent” connection attempts. This could terminate GNUTella connections when they were made. The downside is that a false positive could terminate good traffic. It is also possible for active response from a passive IDS to be ineffective.

Multiple choice test question:

```
11/14/2002 19:46:41.186507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 76:
midgaard.smsc.com.61784 > sehv-3-lt-48.lino.sympatico.ca.8330: P [tcp sum ok]
671312156:671312178(22) ack 16686651 win 16440 (DF) (ttl 123, id 63239, len 62)
0x0000       4500 003e f707 4000 7b06 d8ae aa81 3278  E..>..@.{.....2x
0x0010       8ed9 c430 f158 208a 2803 691c 00fe 9e3b  ...0.X..(.i....;
0x0020       5018 4038 6675 0000 474e 5554 454c 4c41  P.@8fu..GNUTELLA
0x0030       2043 4f4e 4e45 4354 2f30 2e34 0a0a       .CONNECT/0.4..

11/14/2002 19:46:47.176507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 76:
midgaard.smsc.com.61784 > sehv-3-lt-48.lino.sympatico.ca.8330: P [tcp sum ok]
0:22(22) ack 1 win 16440 (DF) (ttl 123, id 63787, len 62)
0x0000       4500 003e f92b 4000 7b06 d68a aa81 3278  E..>.+@.{.....2x
0x0010       8ed9 c430 f158 208a 2803 691c 00fe 9e3b  ...0.X..(.i....;
0x0020       5018 4038 6675 0000 474e 5554 454c 4c41  P.@8fu..GNUTELLA
0x0030       2043 4f4e 4e45 4354 2f30 2e34 0a0a       .CONNECT/0.4..
```

The packets above are:

A. Not related because the IDs are different
B. Both contain 22 bytes of data
C. Are both GNUTella protocol descriptor packets
D. The second packet is a retransmission of the first
E. B and D
F. None of the above

Answer: E, both B and D.

Detect Submitted:

February 15, 2003
http://cert.uni-stuttgart.de/archive/intrusions/2003/02/msg00160.html

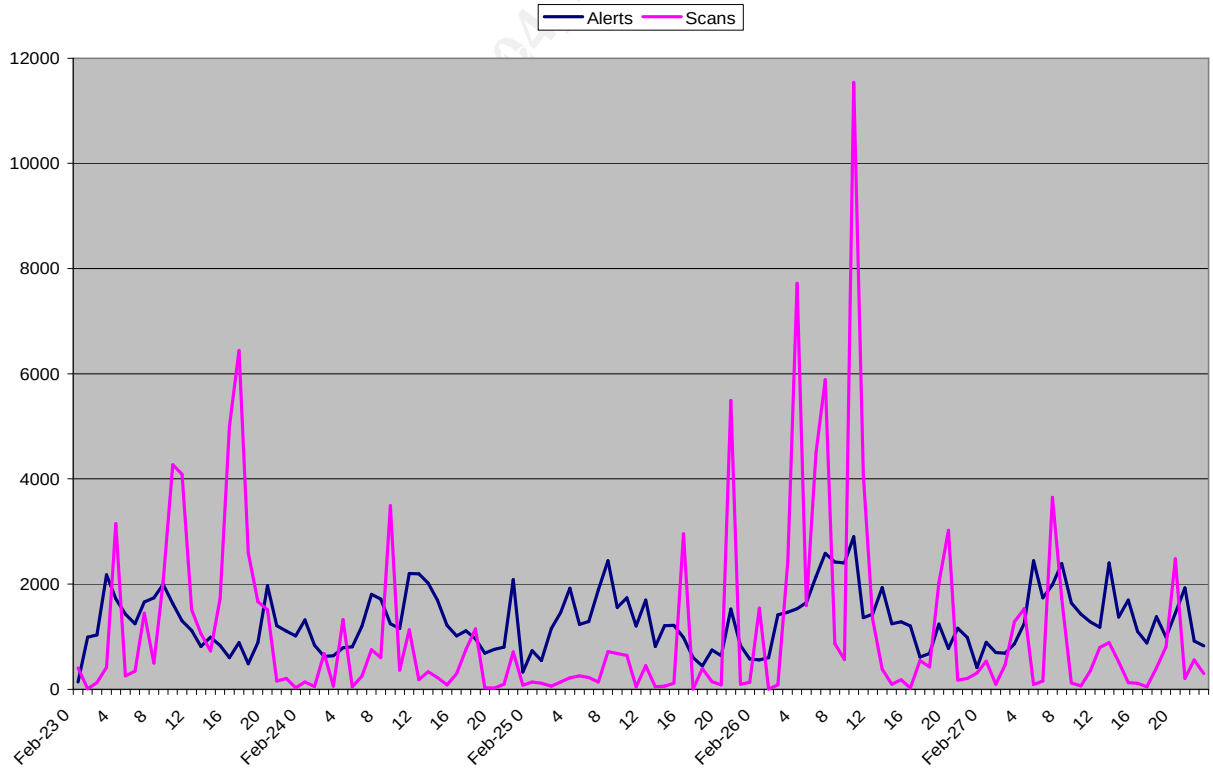
Assignment 3: Analyze This

The purpose of this assignment is to analyze a sequence of intrusion detection log files. The scenario provided is that of a University requesting a security audit. I have taken the approach of a security consultant providing this audit. The remainder of this assignment depicts a report provided to the GIAC University from the fictitious company Shuck Consulting Group (SCG). This assumes that the GCIA Practical assignment part 3 was the original "statement of work".

Executive Summary

A Security Audit is a key component in the overall security of an organization. Shuck Consulting Group (SCG) appreciates the opportunity to complete this audit for GIAC University.

The key mission of a Security Audit is to identify security issues or threats, identify possible vulnerabilities, and recommend safeguards. A threat is any tool or technique that can be used to damage a resource, such as a network, server or desktop, or to compromise those resources for unauthorized use. A vulnerability is the weakness of a resource or procedure that allows a threat to occur. A safeguard is a control or countermeasure employed to reduce the risk associated with a specific threat, or group of threats. The graph below depicts the number and frequency of alert and port scan events detected in the log files.



SCG has completed the Security Audit by conducting an in-depth analysis of log files generated by GIAC University's Snort Intrusion Detection System (IDS). The true IP addresses range of the University was obfuscated in the logs with a prefix of "MY.NET". SCG has converted all obfuscated addresses with this prefix to the unused "10.199" Class B range. This range was chosen at random after verification that the range did not appear elsewhere in the logs, and is used throughout the remainder of this document.

There were 31,760 unique internal (addresses that began with "MY.NET") IP addresses. This is roughly half of the possible 65,534 host for a Class B range. There were 22,209 unique external IP addresses. SCG also discovered 154,160 individual alerts and 132,769 individual port scans in the log file analysis. This consisted of 49 unique alerts and 13 unique scans.

List of Files

The following files were provided by GIAC University. The files were produced by Snort IDS. Although, the exact version and the rules used are unknown, the following can be assumed. The version is at least version 1.9.0 because there are alerts from the 'spp_http_decode' plugin which was added 10/09/2002 or 1.9.0. There were three separate types of files provided. Additional information on how the files were used in the analysis can be found in the "*Description of Analysis Processes*" section.

Alert Files

The following are the "Alert" files. There were 175,478 total lines or records in the alert files. This represented data from 02/23/03 at 00:45:07 to 02/28/03 at 00:05:25. However, there were 212 lines that were corrupted, and were removed.

For example: Lines 4006-4007 in "alert.030223.gz":

4006: `02/23-03:45:35.005242 [**] TFTP - External UDP connection to internal tftp server [**] MY.NET.111.23202/23-03:34:22.324760 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**] MY.NET.251.2:69 -> 192.168.0.253:3724 -> 195.241.150.18:1839`

4007: `:55850`

Filename	Size (in bytes)	Lines / Rec
alert.030223.gz	480,282	35,052
alert.030224.gz	411,336	32,055
alert.030225.gz	427,643	31,915
alert.030226.gz	503,802	38,342
alert.030227.gz	497,009	38,114

The basic format for the Alert files was as follows:

`<date/time stamp> **[**]** Signature **[**]** Source:Port -> Destination:Port`

Scan Files

The following are the "Scan" files that were analyzed. There were 132,769 total lines or records in the scan files. This represented data from 02/23/03 at 00:05:13 to 02/27/03 at 23:46:49.

Filename	Size (in bytes)	Lines / Rec
scans.030223.gz	304,623	39,690
scans.030224.gz	107,409	12,858
scans.030225.gz	96,186	13,351
scans.030226.gz	383,202	49,540
scans.030227.gz	141,365	17,330

The basic format for the Scan files was as follows:

```
<date/time stamp> Source:Port -> Destination:Port <Protocol Info>
```

Out of Spec (OOS) Files

The following are the "Out of Spec" files that were analyzed. There were 99,605 total lines, but only 11,325 unique records in the OOS files. This represented data from 02/22/03 at 00:07:55 to 02/26/03 at 23:49:06.

Filename	Size (in bytes)	Lines / Rec
OOS_Report_2003_02_23_22505.gz	206,824	43,505 / 5,821
OOS_Report_2003_02_24_24091.gz	107,796	11,336 / 1,230
OOS_Report_2003_02_25_11706.gz	158,886	11,644 / 1,154
OOS_Report_2003_02_26_32018.gz	104,638	17,785 / 1,439
OOS_Report_2003_02_27_17540.gz	100,901	15,335 / 1,681

The basic format for the OOS files was as follows:

```
<date/time stamp> Source:Port -> Destination:Port
<IP Header>
<TCP Header / Options>
<Payload>
```

Relational Analysis

One of the first tasks in the analysis process was to determine as much as possible about the University systems. Since no information was provided, SCG has made the following assumptions based on the type and direction of traffic. These assumptions were made based on the origination and receipt of traffic on well known ports. It is important to remember that SCG only had alert and port scan traffic. Inbound traffic was only considered evidence of a service if significant numbers were received.

FTP Servers

To identify possible FTP servers, SCG looked for TCP port 20 and 21 traffic. This identified 3 possibilities. It would be very common for an FTP server to have a lot of port 21 FTP Control traffic. It was also found that all 279 instances of traffic to 10.199.100.165 triggered the "**CS WEBSERVER - external ftp traffic**" signature.

destination	port	count	destination	port	count	destination	port	count
10.199.100.165	21	279	10.199.24.47	21	5	10.199.24.27	21	2

Although, there was very little port 21 traffic to the 10.199.24.47 and 10.199.24.27 addresses, some of this traffic triggered the "**FTP passwd attempt**" signature. This would indicate the presence of an FTP server. Based on this information, there is evidence that all three identified addresses above are indeed FTP servers. There was also evidence that 10.199.208.210 may be an FTP server.

SMTP Servers

In an effort to identify the mail or SMTP servers, SCG searched for multiple instances of traffic to and from port 25. This search uncovered the following 6 unique possibilities. The server 10.199.6.47 has traffic both to and from port 25, so this is most likely an actual SMTP server. The others are less certain.

source	port	count	destination	port	count	destination	port	count
10.199.6.47	25	6	10.199.6.40	25	39	10.199.6.47	25	28
10.199.24.21	25	4	10.199.24.22	25	25			
10.199.12.2	25	2	10.199.24.23	25	31			

There was also the corroboration of some POP v3 traffic to 10.199.25.21. However, there was only a single packet which could have been part of a port scan. Based on this information, it is strongly believed that 10.199.6.47 is the only SMTP server.

DNS Servers

There was very little evidence of Domain Name Service in the log files. The following 4 destinations were discovered with traffic to port 53. However, this was discovered to be NMAP TCP Ping traffic, and as such does not give a strong indication of the presence of an actual DNS server.

destination	port	count	destination	port	count
10.199.1.3	53	22	10.199.137.7	53	2
10.199.1.4	53	2	10.199.1.5	53	1

Based on the available information and the sequential nature of the addresses, SCG believes that 10.199.1.3, 10.199.1.4, and 10.199.1.5 may all be DNS servers.

Network Equipment

Again, SCG was not provided with any information regarding the internal structure of the University. However, one of the traffic patterns that emerged was the use of Trivial FTP. This is very commonly used by network equipment to transfer code and configuration files. It is also a common practice to place network equipment in the upper IP address range. It is the opinion of SCG that these devices are network switches or routers.

source	port	count	source	port	count
10.199.111.235	69	305	10.199.111.231	69	281
10.199.111.230	69	296	10.199.111.219	69	279
10.199.111.232	69	288	10.199.251.2	69	1

All of the traffic from these devices had a single destination of 192.168.0.253.

LDAP Server

Based on the large amount of traffic originating from the following device on port 1760, it is believed that this device is some type of LDAP server.

source	port	count
10.199.244.78	1760	1077

Web Servers

There were significant instances of traffic on the well known HTTP port 80. However, it is also very common to use this port for other purposes in an effort to bypass Firewalls, etc. SCG looked primarily at traffic originating traffic on port 80, and correlated this with traffic destined for port 80. Sources with only a single incident of traffic were eliminated, unless additional information was discovered.

This analysis uncovered the following 6 possible web servers.

source	port	count	destination	port	count
10.199.24.44	80	68	10.199.24.44	80	65
10.199.100.165	80	9	10.199.100.165	80	9816
10.199.179.77	80	4	10.199.179.77	80	10
10.199.6.7	80	4	10.199.6.7	80	211
10.199.218.26	80	3	10.199.218.26	80	847
10.199.24.34	80	3	10.199.24.34	80	19

Although, there were some large amounts of port 80 traffic directed at other devices, this was considered to be coincidental.

destination	port	count
10.199.116.86	80	169
10.199.30.4	80	151
10.199.220.18	80	70
10.199.252.133	80	24
10.199.252.251	80	23
10.199.130.14	80	15
10.199.249.18	80	13

One additional piece of information was uncovered. There were two signatures named "CS WEBSERVER - external ftp traffic" and "CS WEBSERVER - external web traffic". These signature appear to reference a Computer Science Web Server. The destination of all traffic that triggered one of these signatures was 10.199.100.165.

Therefore, SCG is convinced that 10.199.100.165 is the "CS Web Server", and that the other five devices listed in the first table are also hosting web services.

Alerts

The following section covers the in-depth analysis conducted on the alerts, port scans and out of specification events found in the log files supplied. This section is divided into four primary areas:

- **Overview** – which details the type and quantity of alerts discovered
- **Portscan** – which details the type and quantity of scan or reconnaissance activity
- **Out of Spec** – which details the type and quantity of abnormal activity
- **Alert Analysis** – which explains the various alerts and assesses the possible risk to the University

Overview

The following table identifies each unique alert found in the log files. In addition, the number of alerts detected, the number of sources and destinations for each alert, and if the alert was part of the default Snort IDS rule files.

SCG has discovered 154,160 individual alerts in the log file analysis. This consisted of 49 unique alerts. However, of the 49 unique alert signatures, only 7 were an exact match for a Snort IDS rule file signature. Every effort has been made to assure that the evaluation of the alerts was thorough, but some assumptions were necessary since the actual rule or signature files were not supplied.

Default Rule	Signature	# Alerts	# Sources	# Dests
N	SMB Name Wildcard	71720	16168	31107
N	Watchlist 000220 IL-ISDNNET-990517	25714	164	314
Y	spp_http_decode: IIS Unicode attack detected	13375	771	814
N	High port 65535 tcp - possible Red Worm - traffic	11479	145	148
N	CS WEBSERVER - external web traffic	9743	3304	2
N	Port 55850 tcp - Possible myserver activity - ref. 010313-1	4996	80	87
Y	spp_http_decode: CGI Null Byte attack detected	2459	143	96
N	Tiny Fragments - Possible Hostile Activity	1747	12	158
N	SUNRPC highport access!	1663	51	30
N	TFTP - Internal TCP connection to external tftp server	1538	33	35
N	TFTP - External UDP connection to internal tftp server	1449	5	1
N	Null scan!	1416	81	74
N	High port 65535 udp - possible Red Worm - traffic	1202	192	219
N	Watchlist 000222 NET-NCFC	1023	44	33
N	External RPC call	882	4	846
N	Russia Dynamo - SANS Flash 28-jul-00	732	2	3
N	Queso fingerprint	583	231	88
N	10.199.30.4 activity	382	126	1
Y	Incomplete Packet Fragments Discarded	356	53	38
N	TCP SRC and DST outside network	289	32	94
N	CS WEBSERVER - external ftp traffic	279	99	1
N	EXPLOIT x86 NOOP	248	70	73
N	Possible trojan server activity	182	38	114
N	connect to 515 from outside	133	8	3
N	10.199.30.3 activity	102	16	1
N	IRC evil - running XDCC	95	25	18
Y	SNMP public access	94	18	16

Default Rule	Signature	# Alerts	# Sources	# Dests
N	EXPLOIT x86 setuid 0	68	66	59
N	NMAP TCP ping!	57	21	29
N	EXPLOIT x86 stealth noop	41	5	4
N	EXPLOIT x86 setgid 0	32	32	30
N	TFTP - Internal UDP connection to external tftp server	29	13	13
N	SMB C access	7	6	5
N	Probable NMAP fingerprint attempt	7	5	5
N	Port 55850 udp - Possible myserver activity - ref. 010313-1	7	3	3
N	FTP passwd attempt	5	5	2
N	SYN-FIN scan!	4	4	4
N	Attempted Sun RPC high port access	3	3	3
N	Notify Brian B. 3.56 tcp	3	3	1
N	Notify Brian B. 3.54 tcp	3	3	1
Y	Fragmentation Overflow Attack	3	1	1
N	RFB - Possible WinVNC - 010708-1	2	2	2
N	HelpDesk 10.199.83.197 to External FTP	2	1	1
N	IDS552/web-iis_IIS ISAPI Overflow ida nosize [arachNIDS]	1	1	1
Y	NETBIOS NT NULL session	1	1	1
Y	DDOS shaft client to handler	1	1	1
N	FTP DoS ftpd globbing	1	1	1

Portscans

The basic idea behind port scanning is to determine which devices are “listening” or have services running on various ports. Port scanning is a technique often used as a reconnaissance tool, and is typically the precursor to an actual attack or exploit. As such, these events are very serious, and have been carefully evaluated by SCG.

The following table outlines the 132,769 port scan events comprising 13 unique types of scans discovered in the University log files. Note the use of the term “EOI”, this is short for “Events of Interest”.

EOIs by Alert Message		
77669	UDP scan (Externally-based)	
52182	SYN scan (Externally-based)	
1539	NULL scan (Externally-based)	
522	NOACK scan (Externally-based)	
244	INVALIDACK scan (Externally-based)	
170	VECNA scan (Externally-based)	
156	FIN scan (Externally-based)	
122	XMAS scan (Externally-based)	
113	UNKNOWN scan (Externally-based)	
22	NMAPID scan (Externally-based)	
13	FULLXMAS scan (Externally-based)	
9	SPAU scan (Externally-based)	
8	SYNFIN scan (Externally-based)	
Total Uniques:	13	Total EOIs: 132769

There are various methods and techniques for port scanning. Most of the more unusual techniques are geared towards avoiding detection or bypassing filters or Firewalls.

UDP Scan

The basic idea behind a UDP scan is similar to most port scanning. A “probe” or fake UDP packet is sent to the UDP port being scanned. Unlike normal TCP port scanning, no reply is required if a packet is not expected, and no response is required if the device is not listening on that port. However, most devices send an ICMP “Port Unreachable” message if they receive a packet on a closed port. What this means is that UDP port scanning looks for ports that are “closed”, and assumes the other ports are open. This leads to many false positives since there is no guarantee an ICMP or UDP message will be received. The UDP Scan is also quite time consuming. It also typically generates the most alerts. This is caused by the large number of ports to scan and the re-transmission necessary to reduce false positives.

SYN Scan

This is a type of TCP port scanning. It is often called a “half-open” scan. This is because all TCP communication involves the exchange of three packets called the three way handshake. The “SYN” flag is set on the first packet of this exchange. Under normal conditions, if a device is listening on a port, it will respond to a “SYN” packet with a “SYN/ACK” packet. In a SYN Scan, once the SYN/ACK is received, the port is recorded and the three way handshake is not completed. Instead a RST packet is sent indicating a desire to stop communication. This type of scan is used because it is often not recorded by the scanned device.

FIN Scan

This is a type TCP port scanning that only sets the FIN flag. A device that is listening on a port will ignore these packets, but a RST packet is sent if the device is not listening on the port. This method is often used as a more clandestine approach. Most Firewalls are concerned with the SYN packet since this is the packet used to initiate a new TCP session. Many Firewalls and other systems now look for the familiar SYN type packets or scans, but overlook the FIN packets.

SYN/FIN Scan

This is a variation on the FIN Scan. The idea is to exploit the way filtering devices or Firewalls determine which TCP flags are set. The SYN and FIN flags set together are not a valid combination, so many devices do not check for this combination.

XMAS Scan

This is another variation of the FIN Scan. In this technique, the FIN, URGent and PUSH flags are all set in the hopes that the combination of TCP flags will not be recognized.

NULL Scan

This variation on the FIN Scan does not set any of the TCP flags. Again, this combination is often not checked by Firewall and filtering devices.

Invalid ACK Scan

This type of scan involves the use of the ACK TCP flag when not appropriate such as in addition to a SYN or SYN/ACK combination.

Vecna Scan

The Vecna Scan is another variation of different combinations of TCP flags. Specifically, the Vecna Scans use combinations of URG, PUSH, URG/PUSH, FIN/URG, and FIN/PUSH. All of these are meant to bypass Firewalls of filtering devices and provide clandestine operation.

SPAU Scan

This is yet another variation on the use of invalid TCP flag combinations. This particular one utilizes the SYN/PUSH/ACK/URG combination.

NOACK Scan

This is the catch all scan type. This is designed to identify all scans that do not match one of the other known scan types. Specifically, if ACK is not set, and the packet is not a SYN or RST packet and the packet is not a known scan type.

Out of Spec

SCG evaluated various aspects of the out of spec traffic. However, the real question is what the purpose of the OOS traffic is. There are three primary reasons why traffic may be deemed out of spec:

Corrupted traffic

This is as simple as it sounds. It is always possible that a device along the path that a packet travels could corrupt the packet. Remember the whole packet is just a bunch of ones and zeros. Hardware can fail and alter one or more of the bits in a packet.

Unfortunately, this is possible, but not very probable. Packets contain checksums that are designed to identify when packets get changed unexpectedly. SCG did not find any indications of bad checksums in the log analysis. This is not to say corrupted traffic could not have been the case in some instances. It is prudent, however, to assume that a very small percentage would be expected to be the result of packet corruption without any indication of bad checksums.

ECN Traffic

It is also possible that some of the traffic identified as out of spec was in fact utilizing the new ECN standard. ECN or Explicit Congestion Notification is defined by RFC3168[7] and as explained by Tod Beardsley in his GCIA Practical[8], makes use of previously reserved bits in the TCP header. This can often cause an IDS to generate an alert for valid traffic.

However, what Tod did not mention in his practical, is that to utilize this new standard there are components of the IP header that most also be in place. Most notable is the "ECN Capable" bit in the IP header. If the OOS traffic had been legitimate ECN traffic, SCG should have found evidence of this in the IP headers of the OSS traffic.

What SCG found did not support the ECN theory. The vast majority of OOS traffic did not have the ECN Capable bit set in the IP header.

tos	count		tos	count
0x0	10394		0x40	11
0x10	9		0x50	2
0x2	8		0x8	13
0x20	881		0xA0	5

As you can see from the table above, only 8 packets contained a value in the low order bits of the "Type of Service" IP header field. This would have identified the packets as ECN Capable or "congestion experienced". Although not definitive proof, it is strong evidence that the OOS packets were not the result of ECN.

[7] http://www.ietf.org/rfc/rfc3168.txt
[8] http://www.giac.org/practical/Tod_Beardsley_GCIA.doc

Reconnaissance techniques

The final and most likely explanation for the OOS traffic is some type of reconnaissance technique. This is typically in the form of some type of port scanning, Firewall or IDS evasion, or operating system or other types of fingerprinting.

The purpose of OOS traffic in reconnaissance is typically avoidance of detection and possibly circumvention of security controls. The premise is that by creating and sending packets that are invalid or abnormal they may be ignored or not handled correctly.

OOS Characteristics

There are some common trends that SCG investigated. One of the most common is to identify the largest sources. These are listed below. This is sometimes referred to as "Top Talkers".

source	count	source	count
213.37.100.232	4078	148.63.247.216	148
200.167.116.27	196	213.140.9.151	142
212.227.109.38	181	66.140.25.156	114
81.182.14.135	173	213.143.83.172	112
148.64.162.157	153	217.217.14.35	109

Another important characteristic of the OOS traffic is the combinations or patterns used. These patterns can often provide insight into the intended purpose. The following table outlines the primary TCP Flag combinations found in the OOS traffic.

Flag combinations	count
12****S*	9903
****P***	997
********	112
12***R**	60
***A**SF	17

*Pattern 12****S**

Based on the primary destination ports for this particular OOS traffic, SCG believes that most can be attributed to some type of peer-to-peer (P2P) client and attempts at covert port scanning.

Destination port	count	Possible Exploit
443	4114	SSL
4662	1720	eDonkey2000 or Overnet P2P
25	1307	SMTP
6346	1057	Gnutella
80	585	Web Traffic
6011	360	X Windows

*Pattern ****P****

Again, based on the primary destination ports for this particular OOS traffic, SCG believes that most can be attributed to some type of peer-to-peer (P2P) client.

Destination port	count	Possible Exploit
1214	500	Kazaa or Morpheous
80	154	Web Traffic
6011	115	X Windows
8333	66	Unknown
2642	42	Tragic
6346	34	Gnutella

*Pattern ******** | 12***R** | ***A**SF*

No significant pattern of destination port was found for these patterns. These are believed to be the result of NULL Scan, and variations on other port scan traffic.

In addition to the "Top Talkers", SCG also identified the primary destinations or targets of this traffic. This information is useful in the creation of defensive recommendations and strategies. The top or major destinations or targets of OOS traffic are displayed below.

destination	count		destination	count
10.199.207.2	478		10.199.249.134	216
10.199.202.50	357		10.199.240.46	188
10.199.24.21	308		10.199.219.14	186
10.199.6.40	287		10.199.209.210	156
10.199.6.47	286		10.199.24.44	152
10.199.24.23	283		10.199.241.114	148
10.199.189.62	276		10.199.252.122	137
10.199.237.66	266		10.199.100.165	124
10.199.222.98	258		10.199.218.26	117
10.199.220.106	221		10.199.226.98	105

It is also important to identify patterns of OOS communication or traffic. The following table identifies the top pair of communicating devices.

source	destination	count		source	destination	count
200.167.116.27	10.199.249.134	196		81.182.14.135	10.199.222.98	135
212.227.109.38	10.199.209.210	156		213.143.83.172	10.199.252.122	112
148.64.162.157	10.199.240.46	153		217.217.14.35	10.199.237.66	109
148.63.247.216	10.199.241.114	148		209.191.132.40	10.199.250.90	83
213.140.9.151	10.199.202.50	141		195.162.218.167	10.199.202.50	79

Of the out of spec traffic recorded, very little was traffic generated by University devices. In fact, there were only 141 packets generated by University devices of the 11,325 packets of out of spec traffic. The following table outlines the University devices involved as sources.

source	count		source	count
10.199.12.2	7		**10.199.208.230**	**75**
10.199.12.4	**51**		10.199.239.158	3
10.199.194.179	1		10.199.253.2	1
10.199.204.94	3			

Two of the University devices are responsible for most of the internal out of spec traffic. These hosts or devices should be investigated further.

Alert Analysis

SCG conducted as thorough an analysis as possible given the data available. Based on this analysis, SCG has assigned each unique alert a risk level. These risk levels are based on the source of the attack, the seriousness of the exploit, the possibility of false positives, and the number of incidents.

SMB Name Wildcard (Low)

Although this rule name was not an exact match for a current Snort rule, SCG believes the rule was similar to the following arachNIDS[9].

```
alert UDP $EXTERNAL any -> $INTERNAL 137 (msg: "IDS177/netbios_netbios-name-
query"; content: "CKAAAAAAAAAAAAAAAAAAAAAAAAAAAA|00 00|"; classtype: info-
attempt; reference: arachnids,177;)
```

This rule detects NETBIOS name queries. These are typically used as reconnaissance to locate poorly configured NETBIOS shares. All NETBIOS port 137 traffic should be blocked at the Firewall. There were 71,720 instances of this alert affecting 31,107 internal addresses. This is 97% of the internal addresses found in the log files. Thomas Halverson analyzed a similar detect in his IDIC practical[10].

Watchlist 000220 IL-ISDNNET-990517 (High)

This was not a default Snort rule. SCG believes this rule was created to detect traffic to or from the Israeli network range 212.179.0.0 - 212.179.255.255. SCG located 149 various occurrences of this network at SANS alone, such as the practicals by Robert Sorensen[11] or George Bakos[12]. This range is listed in 113,607 reports at the Internet Storm Center at Incidents.org[13].

[9] http://www.whitehats.com/info/IDS177
[10] http://www.giac.org/practical/Tomas_Halvarsson.txt
[11] http://www.giac.org/practical/Robert_Sorensen_GCIA.htm
[12] http://www.giac.org/practical/George_Bakos.html
[13] http://isc.incidents.org/source_report.html?order=&subnet=212

Normally, SCG would only recommend close attention to alerts from this rule. However, the bulk of traffic appears to be some form of P2P software. This poses a significant security risk particularly combined with the nature of the source network.

spp_http_decode: IIS Unicode attack detected (low)

This is a default result from a Snort pre-processor. This particular attack attempts to exploit a flaw in Microsoft IIS web servers[14]. The flaw can allow the bypass of security for directories and files by injecting Unicode characters in the path. Christof Voemel did an excellent job of analyzing this attack in his GCIA practical[15].

SCG is assuming that GCIA University has maintained the security patches on externally accessible devices. If so, this is a harmless attack for patches servers. If not, this is a medium risk and should be addressed quickly. Additionally, the vast majority of this traffic was generated by University devices.

High port 65535 tcp - possible Red Worm – traffic (unknown)

This was not a default Snort rule. SCG believes this rule was triggered by all traffic with a source or destination port of 65535. There were a large number of these alerts and it is very odd to have that many on this port. However, with out payload data SCG can not make an accurate analysis. Bradley Urwiller noted similar traffic in his GCIA practical[16].

CS WEBSERVER - external web traffic (none/low)

This was not a default Snort rule. SCG believes this is a rule designed to watch external HTTP traffic to the Computer Science department web server or servers, since there were two destinations. No correlations were possible without additional information.

Port 55850 tcp - Possible myserver activity - ref. 010313-1 (unknown)

This was not a default Snort rule. SCG can only assume that GIAC University has an application called “myserver” that runs on port 55850. Without further information, SCG is unable to assess risk for this alert. Mark Embrich had a similar analysis in his GCIA practical[17].

spp_http_decode: CGI Null Byte attack detected (low)

This is a default result from a Snort pre-processor. The rule looks for the NULL character in web traffic. The vast majority of this traffic was generated by University devices. More information and payload data would be necessary for a accurate analysis, but SCG feels this traffic is primarily “false positives”. Mark Embrich also analyzed this alert in his GCIA practical[18].

[14] http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms00-057.asp
[15] http://www.giac.org/practical/Christof_Voemel_GCIA.txt
[16] http://www.giac.org/practical/Bradley_Urwiller_GCIA.pdf
[17] http://www.giac.org/practical/Mark_Embrich_GCIA.htm
[18] http://www.giac.org/practical/Mark_Embrich_GCIA.htm

Tiny Fragments - Possible Hostile Activity (low)

Although this rule name was not an exact match for a current Snort rule, SCG believes the rule was similar to the following.

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

This rule looks for fragment packets with a size smaller than 25 bytes. This is higly unusual, since normal fragmentation occurs as a result of MTU size. An MTU of 25 would be highly unlikely. The majority of this traffic was destined for 10.199.246.54. Without additional information it is difficult to accurately assess, but SCG believes this is a low risk. SCG does recommend GIAC University research the fragment capability of their Firewalls, since it could be indicative of a Teardrop attack. This type of traffic was also seen by Dale Ross in his IDIC practical[19].

SUNRPC highport access! (medium)

This was not a default Snort rule. SCG believes this rule was looking for a destination port of 32771. There are several RPC vulnerabilities on this port such as “ttdbserv”, “NFS showmount”, and “RPC port listing”. This alert requires further investigation. Many of the alerts have a source of port 80, and are most likely false positives. However, there are a significant number with a source ports known for Trojans and P2P clients. Joeseph Rach also analyzed this type of traffic in his GCIA practical[20] with similar conclusions.

TFTP - Internal TCP connection to external tftp server (low)

This was not a default Snort rule. SCG believes this rule was looking traffic originating internally with an external destination on TCP port 69. TFTP is not a secure protocol and should not be used for external sites. There are several related Advisories and Vulnerability Notes regarding TFTP at CERT, and 16 items on BugTraq.

TFTP - External UDP connection to internal tftp server (medium)

This was not a default Snort rule. SCG believes this rule was looking traffic originating externally with an internal destination on UDP port 69. TFTP is not a secure protocol and should not be allowed through the Firewall. There are several related Advisories and Vulnerability Notes regarding TFTP at CERT, and 16 items on BugTraq.

Null scan! (low)

This was not a default Snort rule. However, SCG believes this rule was looking for traffic with none of the TCP Flag bits set. Chris Kuethe found similar traffic in his GCIA practical[21]. This is most likely low level reconnaissance traffic and should be blocked by the Firewall.

[19] http://www.giac.org/practical/Dale_Ross_GCIA.htm
[20] http://www.giac.org/practical/Joseph_Rach.html
[21] http://www.giac.org/practical/chris_kuethe_gcia.html

High port 65535 udp - possible Red Worm – traffic (low)

This was not a default Snort rule. SCG believes this rule was triggered by all traffic with a source or destination port of 65535. There were not a large number of these alerts, and over half of the alerts originated from University devices. However, with out payload data SCG can not make an accurate analysis. Bradley Urwiller noted similar traffic in his GCIA practical[22].

Watchlist 000222 NET-NCFC (medium)

This was not a default Snort rule. SCG believes this rule was created to detect traffic to or from the Computer Network Center Chinese Academy of Sciences network range 159.226.0.0 - 159.226.255.255. SCG located 23 various occurrences of this network at SANS alone such as the practicals by Crist Clark[23] and Mark Turkia[24]. This range is listed in 7,359 reports at the Internet Storm Center at Incidents.org[25]. The majority of the traffic appears to be web, e-mail, and eDonkey. Based on this SCG has assessed this alert as medium. This alert should be investigated further.

External RPC call (low)

This was not a default Snort rule. SCG believes this rule looks for all external traffic destined for port 111 or SUN RPC. The bulk of the traffic originated from Korea Network Information Center. This traffic should be blocked by the Firewall. There are several related Advisories and Vulnerability Notes regarding RPC at CERT, and 42 items on BugTraq.

Russia Dynamo - SANS Flash 28-jul-00 (medium)

This was not a default Snort rule. SCG was unable to locate any reference to this rule on SANS. The majority of the traffic was between 10.199.105.204 and 194.87.6.50 which is registered to DEMOS-Online Dialup in Russia. The traffic does not have a concrete pattern. SCG finds no information on this alert, but the internal device should be investigated. In his GCIA practical[26], David Osborn came to a similar conclusion.

Queso fingerprint (low)

This was not a default Snort rule. SCG assumes the rule must look for indication of fingerprinting by the Queso application. The assumption would be that the rule is triggered by some combination of TCP Flags and TCP options. SCG believes the rule is looking for the SYN and Reserved or ECN bits and a window size of 1234. This traffic should be blocked by the Firewall. Information that lead to SCG's conclusions can be found in Tomas Halvarsson's GCIA practical[27].

[22] http://www.giac.org/practical/Bradley_Urwiller_GCIA.pdf
[23] http://www.giac.org/practical/Crist_Clark_GCIA.html
[24] http://www.giac.org/practical/Miika_Turkia_GCIA.html
[25] http://isc.incidents.org/source_report.html?order=&subnet=159
[26] http://www.giac.org/practical/David_Oborn_GCIA.html
[27] http://www.giac.org/practical/Tomas_Halvarsson.txt

10.199.30.4 activity (unknown)

This was not a default Snort rule. SCG believes this rule looks for traffic to or from 10.199.30.4. Based on the traffic, it appears this is a web server running Novell (due to the amount of NCP traffic as well). There is insufficient information to assess risk. No correlations were possible without additional information.

Incomplete Packet Fragments Discarded (low)

This is a default result from the Snort pre-processor "spp_frag2". This pre-processor will alert when all of the fragments of a frame are not received. This could indicate a Teardrop attack, although there were not significant numbers of packets received. There are several related Advisories and Vulnerability Notes regarding Fragmentation at CERT, and 31 items on BugTraq.

TCP SRC and DST outside network (low)

This was not a default Snort rule. SCG believes this rule looks for TCP traffic where both source and destination are external addresses or not "10.199". The majority of this traffic was from 192.168.1.102. SCG assumes this is some type of DMZ address range. Valid detects for this type of traffic could indicate address spoofing of an incorrectly configured router. No acceptable correlations were found for this rule.

CS WEBSERVER - external ftp traffic (low)

This was not a default Snort rule. SCG believes this is a rule designed to watch external FTP traffic to the Computer Science department web server. FTP traffic is not secure and should be avoided. If it is used, make sure the FTP software is current for all patches. There are several related Advisories and Vulnerability Notes regarding FTP at CERT, and 50 items on BugTraq.

EXPLOIT x86 NOOP (low)

Although this rule name was not an exact match for a current Snort rule, SCG believes the rule was similar to the Snort rules for NOOP sleds. This typically looks for some of the various Intel NOOP combinations such as "eb02", "9000", "43", or "61". It is impossible for SCG to assess the risk without access to the payload data. These shellcode rules are very prone to false positives. Todd Garrison did a nice job of analyzing this type of attack in his GCIA practical[28].

Possible trojan server activity (high)

This was not a default Snort rule. However, there are several Snort rules that address the apparent intention of this rule. SCG believes this rule looks for traffic with a source or destination port of 27374. Upon closer analysis, SCG believes 10.199.227.134 may be infected with a Trojan. No correlations were possible without additional information.

[28] http://www.giac.org/practical/Todd_Garrison.html

connect to 515 from outside (low)

Although this rule name was not an exact match for a current Snort rule, SCG believes the rule was similar to the Snort rules for "lpr" or "lpd" vulnerabilities. It is assumed that the rule was looking for external traffic destined for internal devices on port 515. The vast majority of this traffic originated from AT&T dial-up access points in Minneapolis and Washington. This traffic should be blocked at the Firewall. The host 10.199.132.42 should be investigated for possible "lpr" exploits. Robert Sorensen came to similar conclusions in his GCIA practical[29].

10.199.30.3 activity (high)

This was not a default Snort rule. SCG believes this rule looks for traffic to or from 10.199.30.3. Based on the traffic, it appears this is a web server running Novell (due to the amount of NCP traffic as well). Although the purpose of this device is unknown, it appears to have one of the port 6667 Trojans or IRC installed. IRC can be potentially risky and should be avoided if possible. No correlations were possible without additional information.

IRC evil - running XDCC (medium)

This was not a default Snort rule. The rule appears to be targeting IRC on ports 6667, 6668, and 7000. By the title, SCG assumes it is also looking for some characteristic of xDCC file sharing capability in IRC. Use of xDCC is a very dangerous practice and there are 25 University devices that may be using this function. A university administrator named TonikGin has written an excellent paper on protecting against this function[30].

SNMP public access (low)

Although this rule name was not an exact match for a current Snort rule, SCG believes the rule was similar to the following.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 161 (msg:"SNMP public access tcp";
flow:to_server,established; content:"public"; reference:cve,CAN-2002-0012;
reference:cve,CAN-2002-0013; sid:1412; classtype:attempted-recon; rev:4;)
```

The traffic could be either TCP or UDP. This traffic should be blocked at the Firewall. E.A Vasquez noted similar finding in his GCIA practical[31]. Additional information can also be found at CVE in CAN-2002-0012[32] and CAN-2002-0013[33].

[29] http://www.giac.org/practical/Robert_Sorensen_GCIA.htm
[30] http://www.russonline.net/tonikgin/EduHacking.html
[31] http://www.giac.org/practical/EAVazquezJr.html
[32] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0012
[33] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0013

EXPLOIT x86 setuid 0 (low)

Although this rule name was not an exact match for a current Snort rule, SCG believes the rule was similar to the following.

```
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86
setuid 0"; content: "|b017 cd80|"; reference:arachnids,436; classtype:system-
call-detect; sid:650; rev:5;)
```

This rule looks for the pattern "b017cd80". It is impossible for SCG to assess the risk without access to the payload data. These shellcode rules are very prone to false positives. The following SecuriTeam.com URL contains an x86 setuid exploit for Pfinger[34].

NMAP TCP ping! (low)

Although this rule name was not an exact match for a current Snort rule, SCG believes the rule was similar to the following.

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

This rule looks for the signature of an Nmap TCP ping which by default sends a single ACK packet to port 80 with an acknowledgement id of 0. This is a reconnaissance attempt. The traffic should be blocked by the Firewall. E.A Vasquez noted similar finding in his GCIA practical[35].

EXPLOIT x86 stealth noop (low)

Although this rule name was not an exact match for a current Snort rule, SCG believes the rule was similar to the following.

```
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86
stealth NOOP";   content: "|eb 02 eb 02 eb 02|"; reference:arachnids,291;
classtype:shellcode-detect; sid:651; rev:5;)
```

This rule looks for the pattern "eb02eb02eb02". It is impossible for SCG to assess the risk without access to the payload data. These shellcode rules are very prone to false positives. The majority of this traffic was destined for 10.199.24.8 on port 119 (NNTP) from news.ums.edu. This is most likely a false positive. No acceptable correlations were found for this rule.

EXPLOIT x86 setgid 0 (medium)

Although this rule name was not an exact match for a current Snort rule, SCG believes the rule was similar to the following.

```
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86
setgid 0"; content: "|b0b5 cd80|"; reference:arachnids,284; classtype:system-
call-detect; sid:649; rev:5;)
```

[34] http://www.securiteam.com/exploits/6E00Q006AG.html
[35] http://www.giac.org/practical/EAVazquezJr.html

This rule looks for the pattern "b0b5cd80". It is impossible for SCG to assess the risk without access to the payload data. These shellcode rules are very prone to false positives. The majority of this traffic is destined for ports known for P2P software. This is most likely a false positive for a shellcode exploit, but may be P2P traffic. There are 3 articles regarding this exploit in the ISS X-Force database[36].

TFTP - Internal UDP connection to external tftp server (low)

This was not a default Snort rule. SCG believes this rule was looking traffic originating internally with an external destination on UDP port 69. TFTP is not a secure protocol and should not be used if possible. There are several related Vulnerability Notes regarding TFTP at CERT, and 16 items on BugTraq.

SMB C access (medium)

Although this rule name was not an exact match for a current Snort rule, SCG believes the rule was similar to the following.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB C$ ccess";
flow:to_server,established; content: "|5c|C$|00 41 3a 00|"
;reference:arachnids,339; classtype:attempted-recon; sid:533; rev:5;)
```

The rule looks for any NETBIOS access for the 'C' drive. There were 7 alerts involving external access to internal devices. This traffic is very difficult to secure since many Microsoft functions utilize port 139, therefore it should be blocked at the Firewall. Eric Hacker[37] found the same type of detect in his analysis of a site. His recommendations mirrored those of SCG.

Probable NMAP fingerprint attempt (low)

Although this rule name was not an exact match for a current Snort rule, SCG believes the rule was similar to the following.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap fingerprint
attempt"; flags:SFPU; reference:arachnids,05; classtype:attempted-recon;
sid:629; rev:1;)
```

This rule looks for the SYN/FIN/PUSH/URG flags to be set. This is used in OS fingerprinting by Nmap, and is used for reconnaissance. Again, this traffic should be blocked at the Firewall. Chris Kuthe also recorded this signature in his GCIA practical[38].

Port 55850 udp - Possible myserver activity - ref. 010313-1 (low)

This was not a default Snort rule. SCG can only assume that GIAC University has an application called "myserver" that runs on port 55850. Without further information, SCG is unable to assess risk for this alert. No correlations were possible without additional information.

[36] http://www.iss.net/security_center/
[37] http://www.giac.org/practical/Eric_Hacker.html#anchor9566546
[38] http://www.giac.org/practical/chris_kuethe_gcia.html

FTP passwd attempt (none)

This was not a default Snort rule. SCG believes the rule would have been looking for content of "passwd" in FTP (port 21) traffic. SCG is uncertain of the purpose of this rule. In any case, FTP is not a secure protocol as demonstrated by this rule. The password is sent in clear text. SSH should be used instead. There are several related Advisories and Vulnerability Notes regarding FTP at CERT, and 50 items on BugTraq.

SYN-FIN scan! (low)

Although this rule name was not an exact match for a current Snort rule, SCG believes the rule was similar to the following.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN SYN FIN";flags:SF;
reference:arachnids,198; classtype:attempted-recon; sid:624; rev:1;)
```

This is a common scan used to bypass Firewalls and filter devices. It is used for reconnaissance. Herschel Gelman did some excellent analysis of this in his GCIA practical[39].

Attempted Sun RPC high port access (low)

This was not a default Snort rule. However, SCG believes that this rule was designed to trigger an alert on all TCP traffic to an internal device with a destination port or 32771 or greater or possibly 32771 to 34000. This inbound traffic should be blocked at the Firewall. There are several related Advisories and Vulnerability Notes regarding RPC at CERT, and 42 items on BugTraq. Dale Ross also analyzed similar traffic in his GCIA practical[40].

Notify Brian B. 3.56 tcp (medium)

This was not a default Snort rule. The rule appears to trigger on any traffic to 10.199.3.56. The significance of this is unknown. SCG believes that this device is a Window machine running MS-SQL. It also appears to have a Trojan or IRC on port 6667. No correlations were possible without additional information.

Notify Brian B. 3.54 tcp (low)

This was not a default Snort rule. The rule appears to trigger on any traffic to 10.199.3.56. The significance of this is unknown. SCG believes that this device is a Window machine running MS-SQL. Without further information, SCG is unable to assess risk for this alert. No correlations were possible without additional information.

[39] http://www.giac.org/practical/Herschel_Gelman.html
[40] http://www.giac.org/practical/Dale_Ross_GCIA.htm

Fragmentation Overflow Attack (low)

This is a default result from the Snort pre-processor "spp_defrag". This alert is triggered by a packet that has a fragment offset and length that would cause it to extend beyond the previously indicated total packet size. This type of attack is designed to be a denial of service. The only source for this traffic, 80.135.229.168, was a dial-up ISP that is not listed at Dshield.org[41], and there were only 3 instances. There are several related Advisories and Vulnerability Notes regarding Fragmentation at CERT, and 31 items on BugTraq

RFB - Possible WinVNC - 010708-1 (low)

This was not a default Snort rule. SCG believes that this rule was triggered by traffic with a source or destination port of 5900, which is the virtual network computer (VNC) port. SCG feels these 2 alerts were false positives. Obviously, this port should be blocked at the Firewall. Mark Embrich detected similar traffic in his GCIA practical[42]. There are 3 Vulnerability Notes regarding VNC at CERT, and 13 items on BugTraq.

HelpDesk 10.199.83.197 to External FTP (low)

This was not a default Snort rule. The rule appears to trigger on external bound traffic from 10.199.83.197. All of this traffic was destined for a Network Associates IP of 161.69.201.237. No correlations were possible without additional information.

IDS552/web-iis_IIS ISAPI Overflow ida nosize [arachNIDS]

Although this rule name was not an exact match for a current Snort rule, SCG believes the rule was similar to the following.

```
alert TCP $EXTERNAL any -> $INTERNAL 80 (msg: "IDS552/web-iis_IIS ISAPI
Overflow ida"; dsize: >239; flags: A+; uricontent: ".ida?"; classtype:
system-or-info-attempt; reference: arachnids,552;)
```

This actual rule may have had the "dsize" option removed based on the title of the alert. This attack is used to exploit an ISAPI vulnerability in Microsoft IIS web servers. Additional information can be found at CVE in CAN-2000-0071[43] and at BugTraq in BID1065[44]. Bradley Urwiller also detected similar traffic in his GCIA practical[45].

NETBIOS NT NULL session (low)

This was the following default Snort rule.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS NT NULL session";
flow:to_server,established; content: "|00 00 00 00 57 00 69 00 6E 00 64 00 6F
00 77 00 73 00 20 00 4E 00 54 00 20 00 31 00 33 00 38 00 31|"; reference:
bugtraq,1163; reference:cve,CVE-2000-0347; reference:arachnids,204;
classtype:attempted-recon; sid:530; rev:7;)
```

[41] http://www.dshield.org/ipinfo.php?ip=80.135.229.168
[42] http://www.giac.org/practical/Mark_Embrich_GCIA.htm
[43] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0071
[44] http://www.securityfocus.com/bid/1065
[45] http://www.giac.org/practical/Bradley_Urwiller_GCIA.pdf

This rule is looking for Microsoft NULL sessions. These are sessions that do not require any authentication. They are used to perform reconnaissance on a Windows machine. A great deal of information can be obtained if these are allowed. Additional information can be found at CVE in CVE-2000-0347[46] and at Microsoft in KB132679[47].

DDOS shaft client to handler

This was the following default Snort rule.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 20432 (msg:"DDOS shaft client to
handler"; flags: A+; reference:arachnids,254; classtype:attempted-dos;
sid:230; rev:1;)
```

This rule looks for traffic from an external source to an internal destination on port 20432. Since this is an ephemeral port, this rule generates a large number of false positives. The only source of this alert is listed at Dshield.org[48], but since there was only a single packet, SCG believes this was most likely part of a port scan. Chris Calabrese also detected this type of traffic in his GCIA practical[49].

FTP DoS ftpd globbing

Although this rule name was not an exact match for a current Snort rule, SCG believes the rule was similar to the following.

```
alert TCP $EXTERNAL any -> $INTERNAL 21 (msg: "IDS487/ftp_dos-ftpd-globbing";
flags: A+; content: "|2f2a|"; classtype: denialofservice; reference:
arachnids,487;)
```

This rule looks for traffic containing the pattern “2f2a” with a destination port of 21. This is a known exploit of the Wu-FTP server software. If the only destination address, 10.199.208.210, for this alert is running the Wu-FTP server, it should be investigated for possible compromise and latest patches. Mark Embrich found similar traffic in his GCIA practical[50]. Additional information on this exploit can be found at BugTraq in BID3581[51].

Top Talkers

There are many different criteria used to determine a “Top Talker”. SCG utilizes the following criteria to identify the “Top Talkers”. Although it is common and was requested by GIAC University, SCG has identified at least the top ten talkers using the various criteria, but in some cases may identify more than ten. This is because SCG finds that patterns often emerge, and that top talkers often gather in striation patterns.

[46] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0347
[47] http://support.microsoft.com/default.aspx?scid=kb%3Ben-us%3B132679
[48] http://www.dshield.org/ipinfo.php?ip=66.115.47.66
[49] http://www.giac.org/practical/Chris_Calabrese_GCIA.html
[50] http://www.giac.org/practical/Mark_Embrich_GCIA.htm
[51] http://www.securityfocus.com/bid/3581

Also take notice of the additional information at the end of each table. SCG has also included the minimum and maximum number of packets for a given type of "talker", and the average and standard deviation values. These numbers, particularly the standard deviation, help identify how unusual the top talkers are compared to the other talkers of the same type.

Top Alert Destination IP

The following table identifies the top 12 destination IP addresses for Alert traffic.

destination	count	destination	count	destination	count
10.199.100.165	10133	10.199.210.238	1722	24.193.129.45	1150
10.199.240.234	6027	10.199.180.39	1615	67.81.224.77	1077
10.199.247.102	3506	10.199.245.50	1577	10.199.252.122	1045
10.199.244.78	2047	192.168.0.253	1450	64.12.54.248	1033
minimum	**maximum**	**average**	**Std deviation**		
1	10133	4.69	75.12		

Top Alert Source IP

The following table identifies the top 12 source IP addresses for Alert traffic.

source	count	source	count
212.179.94.48	6021	10.199.246.54	1510
212.179.13.98	3502	212.179.100.234	1414
202.175.95.50	2545	212.179.35.118	1252
67.81.224.77	2007	10.199.241.182	1151
10.199.207.34	1714	10.199.244.78	1099
212.179.102.22	1574	10.199.201.66	967
minimum	**maximum**	**average**	**Std deviation**
1	6021	7.045	63.74

Top Alert IP Pairs

The following table identifies the top 10 communication pairs for Alert traffic.

source	destination	count	source	destination	count
212.179.94.48	10.199.240.234	6021	212.179.100.234	10.199.210.238	1413
212.179.13.98	10.199.247.102	3502	10.199.241.182	24.193.129.45	1150
202.175.95.50	10.199.100.165	2545	10.199.244.78	67.81.224.77	1077
67.81.224.77	10.199.244.78	2007	10.199.201.66	80.129.80.137	967
212.179.102.22	10.199.245.50	1574	65.79.79.242	10.199.240.186	787
minimum	**maximum**	**average**	**Std deviation**		
1	6021	1.98	32.02		

Top Alert Destination Ports

The following table identifies the top 12 destination ports for Alert traffic.

destination	count	destination	count	destination	count	destination	count
137	71722	65535	5244	1214	2427	2708	1619
80	27028	6699	4046	1760	2007	3162	1446
2561	6021	55850	2980	32771	1666	4662	1418
minimum	**maximum**	**average**	**Std deviation**				
0	71722	81.46	1798.04				

Top Alert Source Ports

The following table identifies the top 16 source ports for Alert traffic.

source	count	source	count	source	count	source	count
1026	8820	1028	6600	1029	4443	55850	2022
1025	8812	137	6555	2163	3507	1031	1381
1027	7690	3920	6024	1030	2468	1214	1279
65535	7441	80	4928	69	2249	1760	1078
minimum	**maximum**	**average**	**Std deviation**				
0	8820	8.85	167.57				

Top Scan Destination IP

The following table identifies the top 10 destination IP addresses for Scan traffic.

destination	count	destination	count	destination	count
217.210.106.58	10704	212.39.90.9	3291	208.187.180.214	1251
80.232.11.236	6949	217.39.77.130	3052	157.193.80.50	1070
208.48.163.2	3541	217.35.71.32	1521		
66.93.105.211	3449	69.3.248.134	1358		
minimum	**maximum**	**average**	**Std deviation**		
1	10704	2.12	58.93		

Top Scan Source IP

The following table identifies the top 15 source IP addresses for Scan traffic.

source	count	source	count	source	count
130.85.218.62	29337	220.114.0.175	3715	130.85.1.3	2468
130.85.60.16	14909	195.25.165.42	3527	130.85.97.52	2386
130.85.87.44	7442	130.85.88.252	3237	212.4.64.29	2203
130.85.221.110	7146	211.199.143.9	2990	212.4.64.28	2040
130.85.219.170	4658	130.85.150.210	2793	61.221.128.34	1810
minimum	**maximum**	**average**	**Std deviation**		
1	29337	179.42	1322.58		

External Sources

At GIAC University's request, SCG has selected 5 external sources to investigate.

212.179.94.48 - bzq-179-94-48.cablep.bezeqint.net

This address was chosen because it was the top source address in all of the alert log files with 6,021 occurrences. This address is not listed at Dshield.org.

```
inetnum:       212.179.80.0 - 212.179.94.255
netname:       CABLES-CONNECTION
descr:         CABLES-CUSTOMERS-CONNECTION
country:       IL
admin-c:       YK76-RIPE
tech-c:        BHT2-RIPE
status:        ASSIGNED PA
remarks:       please send ABUSE complains to abuse@bezeqint.net
mnt-by:        AS8551-MNT
mnt-lower:     AS8551-MNT
notify:        hostmaster@bezeqint.net
changed:       hostmaster@bezeqint.net 20021029
source:        RIPE
route:         212.179.64.0/18
descr:         ISDN Net Ltd.
origin:        AS8551
notify:        hostmaster@bezeqint.net
mnt-by:        AS8551-MNT
changed:       hostmaster@bezeqint.net 20020618
source:        RIPE
role:          BEZEQINT HOSTMASTERS TEAM
address:       bezeq-international
address:       40 hashacham
address:       petach tikva 49170 Israel
phone:         +972 1 800800110
fax-no:        +972 3 9203033
e-mail:        hostmaster@bezeqint.net
admin-c:       YK76-RIPE
tech-c:        MR916-RIPE
nic-hdl:       BHT2-RIPE
remarks:       Please Send Spam and Abuse ONLY to abuse@bezeqint.net
mnt-by:        AS8551-MNT
changed:       hostmaster@bezeqint.net 20021029
changed:       hostmaster@bezeqint.net 20030204
source:        RIPE
person:        Yuval Keinan
address:       bezeq-international
address:       40 hashacham
address:       petach tikva 49170 Israel
phone:         +972 1 800800110
fax-no:        +972 3 9203033
e-mail:        hostmaster@bezeqint.net
mnt-by:        AS8551-MNT
nic-hdl:       YK76-RIPE
changed:       hostmaster@bezeqint.net 20021215
changed:       hostmaster@bezeqint.net 20030204
```

212.179.13.98 - cablep-179-13-98.cablep.bezeqint.net

This address was chosen because it was the second most frequent source address in all of the alert log files with 3,502 occurrences. This address is not listed at Dshield.org.

```
inetnum:        212.179.13.0 - 212.179.14.255
netname:        MATAV-CABLES
mnt-by:         INET-MGR
descr:          MATAV
country:        IL
admin-c:        MR916-RIPE
tech-c:         ZV140-RIPE
status:         ASSIGNED PA
remarks:        please send ABUSE complains to abuse@bezeqint.net
remarks:        INFRA-AW
notify:         hostmaster@bezeqint.net
changed:        hostmaster@bezeqint.net 20021023
source:         RIPE
route:          212.179.0.0/18
descr:          ISDN Net Ltd.
origin:         AS8551
notify:         hostmaster@bezeqint.net
mnt-by:         AS8551-MNT
changed:        hostmaster@bezeqint.net 20020618
source:         RIPE
person:         Miri Roaky
address:        bezeq-international
address:        40 hashacham
address:        petach tikva 49170 Israel
phone:          +972 1 800800110
fax-no:         +972 3 9203033
e-mail:         hostmaster@bezeqint.net
mnt-by:         AS8551-MNT
nic-hdl:        MR916-RIPE
changed:        hostmaster@bezeqint.net 20021027
changed:        hostmaster@bezeqint.net 20030204
source:         RIPE
person:         Zehavit Vigder
address:        bezeq-international
address:        40 hashacham
address:        petach tikva 49170 Israel
phone:          +972 1 800800110
fax-no:         +972 3 9203033
e-mail:         hostmaster@bezeqint.net
mnt-by:         AS8551-MNT
nic-hdl:        ZV140-RIPE
changed:        hostmaster@bezeqint.net 20021027
changed:        hostmaster@bezeqint.net 20030204
source:         RIPE
```

24.193.129.45 - 24-193-129-45.nyc.rr.com

This address was chosen because it was the top external destination address in all of the alert log files with 1,150 occurrences. It was ranked 9th overall. This address is not listed at Dshield.org.

```
OrgName:    ROADRUNNER-NYC
OrgID:      RRNY
Address:    13241 Woodland Park Road
City:       Herndon
StateProv:  VA
PostalCode: 20171
Country:    US

NetRange:   24.193.0.0 - 24.193.255.255
CIDR:       24.193.0.0/16
NetName:    ROADRUNNER-NYC-3
NetHandle:  NET-24-193-0-0-1
Parent:     NET-24-0-0-0-0
NetType:    Direct Allocation
NameServer: DNS1.RR.COM
NameServer: DNS2.RR.COM
NameServer: DNS3.RR.COM
NameServer: DNS4.RR.COM
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2002-04-05
Updated:    2002-11-25

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

OrgAbuseHandle: ABUSE10-ARIN
OrgAbuseName:   Abuse
OrgAbusePhone:  +1-703-345-3416
OrgAbuseEmail:  abuse@rr.com

OrgTechHandle: IPTEC-ARIN
OrgTechName:   IP Tech
OrgTechPhone:  +1-703-345-3416
OrgTechEmail:  abuse@rr.com
```

67.81.224.77 - ool-4351e04d.dyn.optonline.net

This address was chosen because it was the second most frequent external destination address in all of the alert log files with 1,077 occurrences. This address is not listed at Dshield.org.

```
Optimum Online (Cablevision Systems) NETBLK-OOL-4BLK (NET-67-80-0-0-1)
                                  67.80.0.0 - 67.87.255.255
CustName:   Optimum Online (Cablevision Systems)
Address:    111 New South Road
City:       Hicksville
StateProv:  NY
PostalCode: 11801
Country:    US
RegDate:    2003-01-24
Updated:    2003-01-24
NetRange:   67.81.224.0 - 67.81.239.255
CIDR:       67.81.224.0/20
NetName:    OOL-65ELZBNJ5-0821
NetHandle:  NET-67-81-224-0-1
Parent:     NET-67-80-0-0-1
NetType:    Reassigned
Comment:
RegDate:    2003-01-24
Updated:    2003-01-24
```

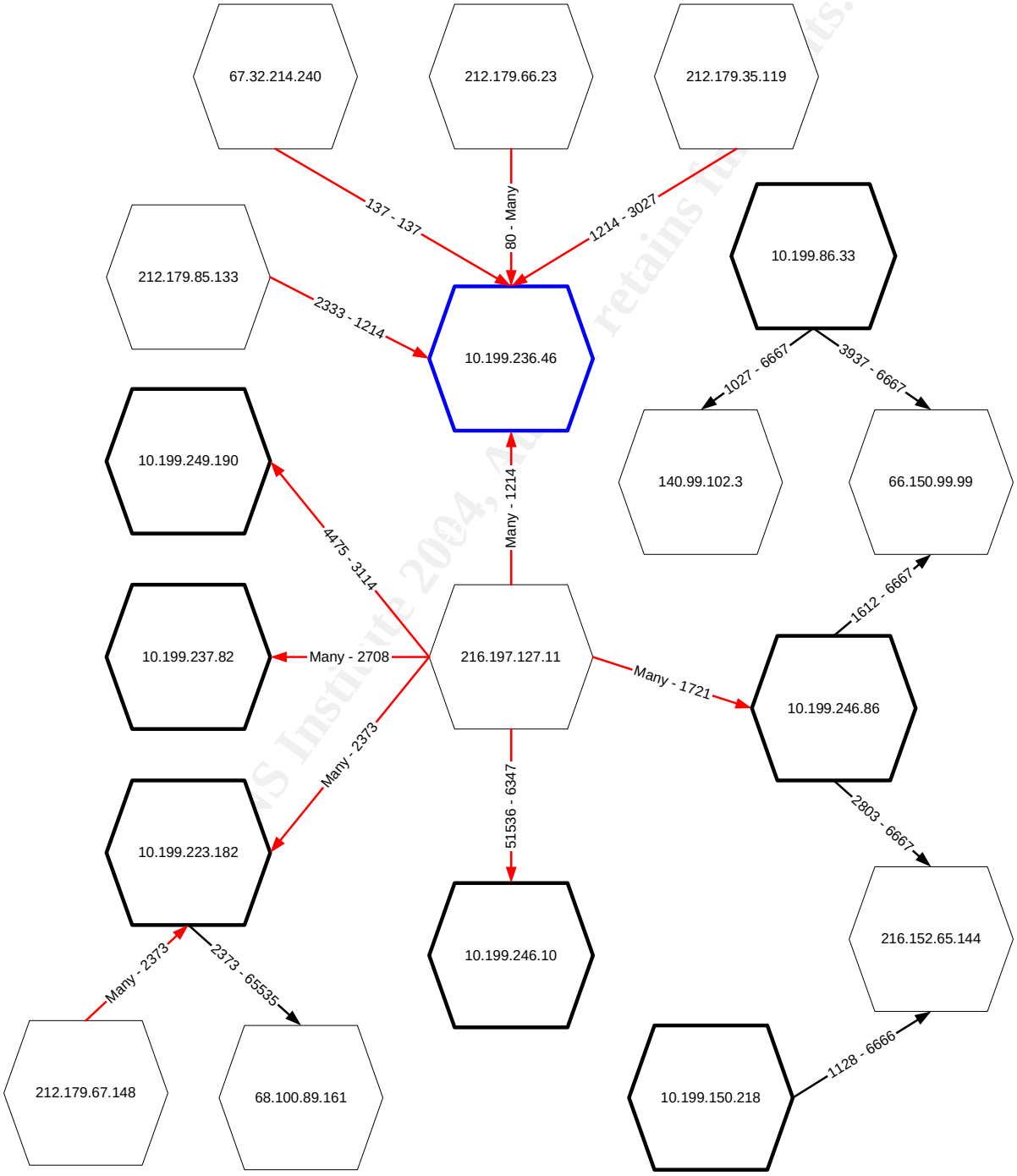
130.85.218.62 - resnet2-89.resnet.umbc.edu

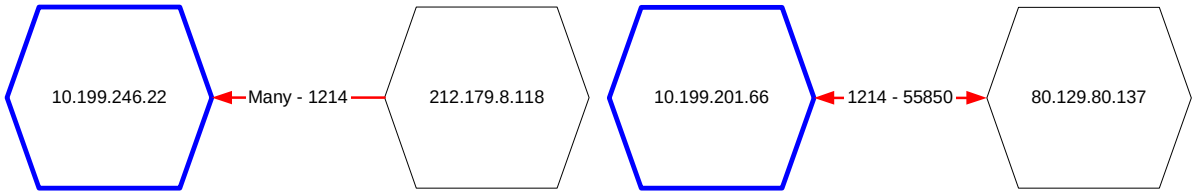
This address was chosen because it was the top source address in all of the scan log files with 29,337 occurrences. This address is not listed at Dshield.org.

```
OrgName:    University of Maryland Baltimore County
OrgID:      UMBC
Address:    UMBC University Computing
City:       Baltimore
StateProv:  MD
PostalCode: 21250
Country:    US
NetRange:   130.85.0.0 - 130.85.255.255
CIDR:       130.85.0.0/16
NetName:    UMBCNET
NetHandle:  NET-130-85-0-0-1
Parent:     NET-130-0-0-0-0
NetType:    Direct Assignment
NameServer: UMBC5.UMBC.EDU
NameServer: UMBC4.UMBC.EDU
NameServer: UMBC3.UMBC.EDU
Comment:
RegDate:    1988-07-05
Updated:    2000-03-17
TechHandle: JJS41-ARIN
TechName:   Suess, John J.
TechPhone:  +1-410-455-2582
TechEmail:  jack@umbc.edu
```

Link Graph Analysis

The following Link Graph depicts a possible KaZaa or other P2P network. To begin, SCG looked for the top talkers on the KaZaa port 1214. Then SQL queries were performed to locate other devices that communicated with our original three talkers (outlined in bold blue). All internal devices are outlined in bold. The links indicate direction of traffic and source and destination ports. All external to internal traffic is highlighted in red.





Notice in the traffic patterns there are several well known Trojan or P2P ports in use.

- 1027 - ICKiller Trojan, 1214 – Kazaa, 2333 - IRC Contact Trojan or SNAPP
- 6347 – Gnutella, 6666 - irc-serv, IRCU, NetBus Trojan, other Trojans
- 6667 – IRC, SubSeven Trojan, other Trojans

It is the opinion of SCG that GIAC University has a large KaZaa or other P2P software user base. This is dangerous software. Its use is typically indicative of other Trojan software. SCG was only able to perform this link graph starting with 3 hosts because addition of more host resulted in a very large link graph.

Possible Compromises / Dangerous Activity

SCG has identified the following possible compromises and dangerous activity.

- Large amounts of P2P software use often to dangerous locations. See destinations of alert “Watchlist 000220 IL-ISDNNET-990517”, and link graph above.
- The following devices may have been exposed to NETBIOS attacks or data theft. 10.199.132.42, 10.199.132.43, 10.199.137.46, 10.199.190.93, 10.199.190.100.
- Significant traffic, 5,026 alerts affecting 226 internal hosts, on several well known Trojan and P2P ports (1025, 1027, 1029,1042,1045, 1097, 1104,1214, 1338, 2333, 4662, 6346,6347, 6666, 6667, 27374, 65535)
- IP 10.199.105.204 may be infected with Russia Dynamo.
- IP 10.199.30.3 may be running IRC or be infected with a Trojan.
- IPs 10.199.12.4 and 10.199.208.230 generated a large amount of OOS traffic.
- IP 10.199.227.134 appears to be infected with a port 27374 Trojan.
- IP 10.199.132.42 may have been compromised via an “lpr” exploit.
- There are 25 University devices that may be using the xDCC file sharing capability of IRC. Check the destinations of alert “IRC evil - running XDCC”.
- There are several devices that may have been exploited via SUN RPC. Check destinations of alert “SUN RPC highport access”.

Defensive Recommendations

The following recommendations are made with the caveat that SCG did not have full access to payload data, or any previous knowledge of the layout of the University.

- Block all external access to NETBIOS port 139 at Firewall.
- Address P2P software issues on destination addresses outlined above.
- Maintain current patch level on all servers, especially externally accessible ones.

- Research the fragment capability of all GIAC University Firewalls.
- Block all incoming high TCP port access at the Firewall, particularly port 32771 and 111.
- Investigate possible use of MonkeyCom (port 9898)
- Discontinue use of TFTP for internal / external traffic. Instead use SSH.
- Verify SUN RPC (port 111) is blocked by the Firewall.
- Block all inbound connections except those explicitly allowed by the University. It is very common for a university to have an "open network" policy. If this is the case, then SCG recommends a minimum of two full time intrusion detection analysts be employed to monitor the network. It is apparent that this is not the case currently.
- Have a qualified third party firm to perform a Vulnerability Assessment of the University network.
- Implement policies and procedures to maintain patch levels on all servers.
- Review the Top Talkers lists and investigate each internal device for possible compromise or mis-configuration. Ideally, you could compare the results of an external IDS to that of an internal one.

Description of Analysis Processes

The following outlines the process SCG employed to analyze the log files supplied by GIAC University. The analysis was conducted with the aid of several tools and operating systems. The bulk of the raw file manipulation was done in Red Hat Linux and under Cygwin on Windows XP. The data was analyzed using Silicon Defense's SnortSnarf[52], scripts found in Tod Beardsley's GCIA practical[53], and various SQL scripts in a Microsoft Access database.

- The first step was getting some record counts from the various files. This was done using '`wc -l`' and for the OOS file '`grep -e "->" OOS_file | wc -l`'.
- Then the 5 daily files were combined into a single file using '`cat`'.
- There were several of the alert files that had corrupted data. It looked as though multiple alerts were written to the same line. These lines were removed by hand using a search for lines that did not start with a date.
- The files were then sorted by date and time using the 'sort' command.
- Many of the analysis tools require valid IP addresses, so the obfuscated "MY.NET" was replaced with an unused private address, "10.199". This was done by making sure 10.199 was unused, `grep -e '10\.199'`. Then replacing the "MY.NET", `sed -e 's/MY.NET/10.199/g'`.
- Then the alert and scan files were analyzed with SnortSnarf.
- Next, the scan and alert files were formatted in a comma separated format to allow import into a database. This was done with the `csv.pl` script by Tod Beardsley[54]. The CSV files were then analyzed with the `summarize.pl` script also by Tod. This script summarized the top talkers and top ports, etc.

[52] http://www.silicondefense.com/software/snortsnarf/
[53] http://www.giac.org/practical/Tod_Beardsley_GCIA.doc
[54] http://www.giac.org/practical/Tod_Beardsley_GCIA.doc

- The OOS files required a little manipulation before they could be imported into the database. First the IP address data was extracted:
```
grep -e '->' OOS_Report-all | awk -F ' ' '{print $2","$4}' |
sed 's/:/,/g' | sed 's/MY.NET/10.199/g' > OOS_Report-all-ip-port.csv
```
- Then the TCP flag data was extracted:
```
grep -e 'Seq:' OOS_Report-all | awk '{print $1","$3","$5","$7","$9}' >
OOS_Report-all-flags.csv
```
- Then the IP header data:
```
grep -e "^TCP TTL" OOS_Report-all| awk '{print $2":"$3":"$4":"$5":"$6}' |
awk -F ':' '{print $2","$4","$6","$8","$10}' > OOS_Report-all-header.csv
```
- The IP and TCP flags were combined in a file for import as well.
```
egrep "^02/|Seq:" OOS_Report-all | sed –e 's/MY.NET/10.199/g' >
OOS_Report-all-foo
```
- The combined file was edited with an editor and made into a CSV record with the regular expression functions.
  REGEX "^([0-9]:^)^([0-9]+^)^(^p^)" REPLACE with "^1^2,"
  REGEX " " REPLACE with ","
  REGEX " " REPLACE with ","
- The CSV files were imported into Microsoft Access where various SQL queries were used. The information generated by SnortSnarf and the other tools were utilized together to fully analyze the data.

References

Goldman, Jeff. "Intrusion Detection Systems: SHADOW." May 2002.
URL: http://www.isp-planet.com/services/ids/shadow.html (22 Feb. 2003).

Beardsley, Tod. "GCIA Practical Assignment." May 2002.
URL: http://www.giac.org/practical/Tod_Beardsley_GCIA.doc (29 Mar 2003).

Halversson, Tomas. "IDIC Practical Assignment." April 2000.
URL: http://www.giac.org/practical/Tomas_Halvarsson.txt (29 Mar 2003).

Sorensen, Robert. "GCIA Practical Assignment." February 2001.
URL: http://www.giac.org/practical/Robert_Sorensen_GCIA.htm (29 Mar 2003).

Bakos, George. "GCIA Practical Assignment." 2000.
URL: http://www.giac.org/practical/George_Bakos.html (29 Mar 2003).

Voemel, Christof. "GCIA Practical Assignment." September 2001.
URL: http://www.giac.org/practical/Christof_Voemel_GCIA.txt (29 Mar 2003).

Urwiller, Bradley. "GCIA Practical Assignment." April 2002.
URL: http://www.giac.org/practical/Bradley_Urwiller_GCIA.pdf (29 Mar 2003).

Embrich, Mark. "GCIA Practical Assignment." February 2002.
URL: http://www.giac.org/practical/Mark_Embrich_GCIA.htm (29 Mar 2003).

Ross, Dale. “GCIA Practical Assignment.” 2000-2001.
URL: http://www.giac.org/practical/Dale_Ross_GCIA.htm (29 Mar 2003).

Rach, Joseph. “GCIA Practical Assignment.” 2000.
URL: http://www.giac.org/practical/Joseph_Rach.html (29 Mar 2003).

Kuthe, Chris. “GCIA Practical Assignment.” Unknown.
URL: http://www.giac.org/practical/chris_kuethe_gcia.html (29 Mar 2003).

Clark, Crist. “GCIA Practical Assignment.” 2000-2001.
URL: http://www.giac.org/practical/Crist_Clark_GCIA.html (29 Mar 2003).

Turkia, Miika. “GCIA Practical Assignment.” January 2001.
URL: http://www.giac.org/practical/Miika_Turkia_GCIA.html (29 Mar 2003).

Oborn, David. “GCIA Practical Assignment.” Unknown.
URL: http://www.giac.org/practical/David_Oborn_GCIA.html (29 Mar 2003).

Garrison, Todd. “GCIA Practical Assignment.” Unknown.
URL: http://www.giac.org/practical/Todd_Garrison.html (29 Mar 2003).

Vazquez, E.A., Jr. “GCIA Practical Assignment.” Unknown.
URL: http://www.giac.org/practical/EAVazquezJr.html (29 Mar 2003).

Hacker, Eric. “GCIA Practical Assignment.” Unknown.
URL: http://www.giac.org/practical/Eric_Hacker.html#anchor9566546 (29 Mar 2003).

Gelman, Herschel. “GCIA Practical Assignment.” 2000.
URL: http://www.giac.org/practical/Herschel_Gelman.html (29 Mar 2003).

Calabrese, Chris. “GCIA Practical Assignment.” December 2001.
URL: http://www.giac.org/practical/Chris_Calabrese_GCIA.html (29 Mar 2003).

Ramakrishnan, K. “The Addition of Explicit Congestion Notification (ECN) to IP.” Request for Comments 3168. September 2001
URL: http://www.ietf.org/rfc/rfc3168.txt (29 Mar 2003).

Unknown, A.K.A TonikGin. “XDCC – An .EDU Admin’s Nightmare” September 2002.
URL: http://www.russonline.net/tonikgin/EduHacking.html (29 Mar 2003).

Last Updated: May 28th, 2017

Upcoming Training

Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANSFIRE 2017 - SEC503: Intrusion Detection In-Depth	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201709,	Sep 11, 2017 - Oct 18, 2017	vLive
Baltimore September 2017 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Scottsdale SEC503	Scottsdale, AZ	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Boston SEC503	Boston, MA	Oct 09, 2017 - Oct 14, 2017	Community SANS
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced