Global Information Assurance Certification Paper

Intrusion Detection In Depth

GCIA Practical Assignment

Version 3.0

Dan G. Hawrylkiw

SANS Network Security
San Diego 2001
October 16th – October 21st, 2001

Table of contents

Introduction

Logging Conventions used in this Paper

The following were used for network detects.

Snort v1.8.1– The Open Source Intrusion Detection System (http://www.snort.org):
This is the NIDS software used to detect and log the events into the MySQL DB.

ACID v0.96b13– Analysis Console for Intrusion Databases (http://www/cert.org/kb/acid):
This is the front-end web interface used to access the Snort data logged in MySQL

ACID extracts are displayed slightly different than default Snort dumps, so an brief line-by-line explanation is included. This depicts a packet from an FTP login

```
#(1 - 733) [2001-11-17 11:23:32]  (138)Unknown Sig Name
(Sensor number, serialized packet ID), [Date Time], signature message
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
Layer 3 protocol, Source IP (obscured by XX's) -> Destination IP
hlen=5 TOS=16 dlen=88 ID=538 flags=0 offset=0 TTL=64 chksum=43235
Header Length, Type of Service, ID, IP flags, Time-To-Live, and hdr
checksum
TCP:  port=21 -> dport: 4672  flags=***AP*** seq=1091630342
Later 4 protocol, source port, destination Port, Flagged bits, sequence
number
ack=2231854025 off=5 res=0 win=32120 urp=0 chksum=50541
Acknowledgement number, layer 4 offset, reserved bits, window size,
Urgent pointer, checksum
Payload:  length = 48
Payload length followed by hex and ASCII payload dump.
000 : 32 33 30 20 47 75 65 73 74 20 6C 6F 67 69 6E 20   230 Guest login
010 : 6F 6B 2C 20 61 63 63 65 73 73 20 72 65 73 74 72   ok, access restr
020 : 69 63 74 69 6F 6E 73 20 61 70 70 6C 79 2E 0D 0A   ictions apply...
```

Fast logging was used in assignment 3. This logging is brief, and does not include protocol, flag, sequence, or payload information. The following format was used:

```
11/08-04:31:54.919725 [**] SMB Name Wildcard [**] 216.150.152.145:137 -
> EDU.NET.5.44:137
Date-Time [**] Rule message (label)[**] source IP:port -> Dest IP/Port
```

Brief Information about my network

The Snort detects were captured from a network that hosts a small website. Access to the Internet is provided via a DSL line with 8 IP addresses available. There are two physical servers (web and mail), serving 6 domains. Both servers share the same Internet facing IP address through a port-forwarding Linux IPChains firewall. Both machines are running custom installations of RedHat Linux 7.1 with patches installed to applicable services. Snort monitors the site from the Internet facing interface of the firewall and logs to a MySQL database.

An occasional honeypot has been placed behind the firewall to gain experience in monitoring black-hat activity. All honeypots are port forwarded and have outbound traffic filtered to protect the Internet community from a compromised machine. All traffic to the honeypots is binary logged by TCPdump and post-processed by snort.

Assignment 1 - Describe the State of ID

Network Intrusion Detection and Use of Automated Response:
Should we use a detective tool as a corrective control?

I overheard a whispered conversation at the San Diego SANS Network Security 2001 Conference that started after David Hoelzer reviewed the signatures of the December 25th, 1994 attack on Tsutomu Shimomura's home network. As the presenter started the review, we were asked to remember "16 seconds". Only after reviewing the signatures and results of the attacks that were used to gain root access to one of Shimomura's workstations, did he explain that the event unfolded in only 16 seconds. The first whisper stated, "An analyst sitting at a console wouldn't be able to stop that attack". The reply was, "He was out of town; there was no way to prevent it anyways". The conversation touched on automated response, but none of the parties involved could agree on the practicality of automating responses to the attack that was, at the time, only theoretical. At that date- practical responses did not exist. And today, the practicality of automating responses still isn't clear.

Introduction

Network Intrusion Detection is the art and science of monitoring networks for activity that may jeopardize the security of the infrastructure under surveillance. By definition and design, this is a detective tool that improves upon the tedious review of logs or recorded traffic by sniffing network traffic and filtering for specific events, typically in real-time. When suspicious traffic is detected, the minimum actions of the Network Intrusion Detection Systems (NIDS) are to log the event and/or contact the appropriate personnel.

While Intrusion detection systems reduce the time it takes to identify suspicious activity, further actions have been dependent on human intervention to begin a response for three reasons. First, Network Intrusion Detection Systems are imperfect and can alert on non-malicious traffic, resulting in false positives. Second, not all legitimate alerts warrant a response. Third, most alerts that warrant a response, require human judgment to determine the most appropriate action. Therefore an analyst is required to further validate alerts and decide if, and how, to take any actions. Unfortunately, the human interaction is the most time consuming element in an attack response cycle.

Modern NIDS can initiate responses in addition to simple notifications. These responses usually fall under direct intervention or scripted reconfiguration of surrounding equipment. An automated response does not necessarily need to address the traffic directly, but could assist the engineers in handling incidents with greater efficiency.

Types of responses

Session Sniping

Direct intervention to disrupt communications between an attacker and victim is often called session sniping or knockdown. This action is performed by injecting packets to break down a connection that triggered the response. The most effective knockdown for a TCP connection is to forge packets to reset the connection. To do this, the NIDS must forge packets to send to one or both systems with the TCP Reset bit set. The source IP, Ports, and sequence numbers must be in sequence with the traffic that triggered the event for the reset to be effective. This response is integrated with or in development for IDS systems that are currently on the market today.

Injecting resets cannot guarantee that the session knockdown will be successful. One reason is that the TCP/IP stacks of the victim and attacker systems may handle the forged resets in different ways. Some TCP/IP stacks will only accept the first packet with the correct sequence number, while ignoring any replacements. Others may overwrite the previously received packets with the latest packet matching the sequence. Because the forged packets are injected into the network in sequence (or in the next sequence) and may be competing with additional traffic from the same session, one cannot assume that the reset packet will be heard by the OS.

ICMP Messaging
Protocols such as UDP cannot be knocked down by sending resets because of their connectionless nature. Since ICMP and UDP do not support transport layer flags to close connections, malicious traffic cannot be stopped by injecting packets without involving higher layer controls. To accommodate this, without forging packets with higher layer payloads, ICMP error messaging can be used. This response sends an ICMP error message to the attacker identifying that the victim, the victim's network, or the destination port is not available. The intention is to have the attacking host believe that the victim cannot be reached, regardless of any other traffic received.

ICMP messaging seeks to notify the attacker's TCP/IP stack that the victim in unavailable in some way. Unfortunately, the likelihood that this message will be received, understood, and followed by the attacking machine is low. Many network attack tools do not use the operating system's TCP/IP stack, and definitely aren't written to be RFC friendly. This extends to the simple hackers attacking from home with consumer equipment. Most home and small business firewalls, whether host or appliance based, don't bother with ICMP error messaging and drop the message on ingress. This increases the likelihood that the ICMP message will be ineffective in stopping an attack.

Shunning
Shunning is the denial of access to a host suspected of originating an attack. Access can be denied at a target host or at a network chokepoint; and can block hosts, networks, or connectivity to specific services. A common response is to block access to an attacker's

IP at an ingress point to reduce the possibility of expanding the attack to other targets within the protected environment.

Firewalls are an ideal location to deny access to a site. The use of scripting or plug-ins such as Checkpoint's Open Platform for Security (OPSEC) allow for simplified NIDS/firewall integration by allowing the NIDS to directly edit the firewall rules. This allows the perimeter access controls to be updated in near-real time by the triggering of events on the Network Intrusion Detection System.

While this may sound ideal, shunning is often avoided because it provides additional control to the attacker. An attacker who identifies a site that is shunning suspicious sources may decide to send forged packets, forcing your firewall to deny services to legitimate users. This was defined by Marty Roesch, author of the Snort NIDS, as "putting the attacker in control of the firewall".

Non-Blocking Responses

The most frequently discussed active responses are blocking countermeasures that intervene directly with either the traffic or the path taken by the traffic. However, non-blocking responses can also be used to protect a computing environment. Most of these responses are innocuous and may seem transparent to the suspect user/system.

Post attack cleanup is possible with the assistance of a NIDS. This response would involve a scripted action to execute upon detection of an attack. While it is unlikely that this could be used to protect from an attacker manually breaching a system and using a variety of tools in infinite number of ways, these scripted responses could address well-known attacks with well known fixes. For example, one of Nimda's attack vectors is to create .eml files on shares, expecting that an unsuspecting victim will attempt to open them with a Windows system. The NIDS, upon detection of this activity, could launch a script to delete the .eml file from the share, scan the share for more suspicious files, and possibly initiate a virus scan on the infected host.

Another non-blocking response that is commonly associated with shunning is redirection. This can be used to protect networks by redirecting suspicious hosts through additional security controls or by changing destinations altogether. In cases where it is not practical to shun hosts, it may be reasonable to redirect attackers through alternate firewalls with more restrictive ACLs. One interesting derivative of redirection is transferring the attacker to a honeypot. This can allow the security administrator to monitor attacks in detail- allowing the attacker to further attack the system without risking production servers.

Extended Notification

One of the basic expectations of a Network Intrusion Detection System is that it must be able to quickly filter and log events of interest. In the interest of reducing response time, most systems have a mechanism to "push" notifications to NIDS console administrators or analysts. While the basic capability might not be considered an active response, extending the notification to identify and contact external entities is an option that could

be scripted. While most contacts should be reviewed first, well-known attacks could be automatically escalated to the attacker's ISP.

Risky Business
The ability to automatically attack back exists, but is not listed as a legitimate response because far too many pitfalls exist for this to be a real-world option outside of a laboratory. Because a NIDS, or human in some cases, cannot quickly discriminate a true attacker from a backdoored host or spoofed traffic source, the possibility of attacking an innocent system exists. Along the potential legal pitfalls of attacking the suspected host, legal precedent exists stating that carriers can hold the "original victim" responsible for their counteractions and potential effect on the carrier's systems. Aside from the issues of counter attack; this is also an ideal way to tip of an attacker, taint evidence, and ruin an investigation.

Food for Thought

While active responses sound great on paper and make great sales pitches, they are not bulletproof solutions. The risk of denying legitimate access to services increases with the use of active responses, especially persistent countermeasures. Automated responses may only work effectively in specific scenarios or on limited traffic types.

Worm and Virus Attacks

The recent success of worm attacks has clearly identified that, in some cases, requiring any human interaction is not fast enough to successfully contain malicious traffic. The infection rates of worms such as CodeRed and Nimda allow the worms to propagate throughout a network in seconds or minutes, shifting the response from containment to cleanup.

Because of the mechanism of infections, session sniping is usually ineffective in stopping these attacks because of the timing and small number of packets involved. If the buffer overflow or command is issued in a single packet, the NIDS will not be able to intervene before the damage is done. For example Appendix A identifies a packet trace from a simulated worm attack that failed to intervene. Additionally the practicality of resetting sessions at a rate of hundreds or thousands of packets per second is likely to overwhelm the NIDS.

Fast, automated responses can help to contain the outbreak. The most effective approach is to "reach ahead" of the attack, by creating network ACLS that block the suspicious traffic or to drop the infected host off the network (perhaps applicable to clients only). This is far less intrusive than shutting down the network or knocking critical servers off the network. For this, signatures can be used to identify well-known attacks, and new attacks could be identified by excessive scanning or expected traffic patterns.

A key requirement for using an automated response to contain an outbreak is assessing the risks involved. An automated response is practical only when the risk of denying legitimate services is far outweighed by the risk of infection, damage, and loss of service.

Limited Shunning

In open networks, such as Universities, where the networks are not isolated from the Internet by a firewall; the networks are an open playground for hackers. Not only are the typical services (Web, SMTP, FTP, etc) available, but all ports and protocols are exposed to attack or misuse. If these networks utilize shunning, similar to a firewalled site, they still give too much control of their perimeter to the attackers.

If shunning is used, it could be restricted to non-standard traffic. This is much less likely to allow the attacker to initiate denial of service attacks by triggering block/shun responses, and still blocks a majority of the exposure points.

Conclusion

Automated responses are a powerful extension to the art of Network Intrusion Detection. They allow the systems to respond to incidents much faster than any human could, by following prescribed steps when triggered.

The risks associated with automated response can be extremely high if not configured correctly. The existence of vague rules and false positives dictates that automated responses are not appropriate for most traffic. Specific traffic types can be addressed, but the risks of denying access to legitimate services must be reduced below the risk of loss from an attack. For this to be true, the Intrusion analysts must have a firm grasp upon the signatures used to identify events. As with any major change, proper review and testing should be done to reduce the possibility of interruption, failure, or escalation of the event.

Appendix A. – Failed TCP Reset Attempt

The following is an example of a TCP reset response issued by a Snort 1.8.3 NIDS after triggering on a "WEB-IIS ISAPI .ida? request". The alert, and subsequent response, is triggered upon detecting the string ".ida?" in the HTTP GET request from the attacker.

The output was generated from TCPDUMP 3.6.2. The attacker's IP is 10.0.0.10, and the victim is at 192.168.1.10. Blue packets indicate traffic between the attacker and the victim. Red Packets were generated by Snort in response to the simulated attack, and only attempt to reset the connection at the victim. The 3-way handshake has been removed for brevity. The simulated attack is executed in packet #4.

```
#4
05:23:15.113298 0:10:67:0:85:7a 0:90:27:8c:e9:f3 ip 536:
10.0.0.10.58702 > 192.168.1.10.http: P 4211792473:4211792955(482) ack
3225804639 win 17520 (DF) (ttl 51, id 11759, len 522)
0x0000       4500 020a 2def 4000 3306 cf19 8fb7 980a  E...-.@.3.......
0x0010       d0ba 5069 e54e 0050 fb0a da59 c045 df5f  ..Pi.N.P...Y.E._
0x0020       5018 4470 9648 0000 4745 5420 2f64 6561  P.Dp.H..GET./dea
0x0030       6675 6c74 2e69 6461 3f4e 4e4e 4e4e 4e4e  fult.ida?NNNNNNN
```

```
0x0040       4e20 4854 5450 2f31 2e30 0d0a 4163 6365  N.HTTP/1.0..Acce
0x0050       7074                                     pt

#5
05:23:15.123298 0:90:27:8c:e9:f3 0:10:67:0:85:7a ip 60:
192.168.1.10.http > 10.0.0.10.58702: . [tcp sum ok]
3225804639:3225804639(0) ack 4211792955 win 31638 (DF) (ttl 63, id
59367, len 40)

#6
05:23:15.123298 0:90:27:8c:e9:f3 0:10:67:0:85:7a ip 572:
192.168.1.10.http > 143.183.152.10.58702: P 3225804639:3225805157(518)
ack 4211792955 win 32120 (DF) (ttl 63, id 59368, len 558)
0x0000       4500 022e e7e8 4000 3f06 08fc d0ba 5069  E.....@.?.....Pi
0x0010       8fb7 980a 0050 e54e c045 df5f fb0a dc3b  .....P.N.E._...;
0x0020       5018 7d78 2a7f 0000 4854 5450 2f31 2e31  P.}x*...HTTP/1.1
0x0030       2034 3034 204e 6f74 2046 6f75 6e64 0d0a  .404.Not.Found..
0x0040       4461 7465 3a20 5375 6e2c 2030 3220 4465  Date:.Sun,.02.De
0x0050       6320                                     c.

#7
05:23:15.123298 0:90:27:8c:e9:f3 0:10:67:0:85:7a ip 60:
192.168.1.10.http > 143.183.152.10.58702: F [tcp sum ok]
3225805157:3225805157(0) ack 4211792955 win 32120 (DF) (ttl 63, id
59369, len 40)

#8
05:23:15.123298 0:90:27:8b:17:17 0:90:27:8c:e9:f3 ip 60:
10.0.0.10.58702 > 192.168.1.10.http: R [tcp sum ok]
4211792473:4211792473(0) ack 3225805121 win 0 (ttl 255, id 22081, len
40)

#9
05:23:15.123298 0:90:27:8b:17:17 0:90:27:8c:e9:f3 ip 60:
10.0.0.10.58702 > 192.168.1.10.http: R [tcp sum ok]
4211792473:4211792473(0) ack 3225805121 win 0 (ttl 255, id 22081, len
40)

#10
05:23:15.243298 0:10:67:0:85:7a 0:90:27:8c:e9:f3 ip 60: 10.0.0.10.58702
> 192.168.1.10.http: . [tcp sum ok] 4211792955:4211792955(0) ack
3225805158 win 17520 (DF) (ttl 51, id 11796, len 40)

#11
05:23:15.253298 0:10:67:0:85:7a 0:90:27:8c:e9:f3 ip 60: 10.0.0.10.58702
> 192.168.1.10.http: F [tcp sum ok] 4211792955:4211792955(0) ack
3225805158 win 17520 (DF) (ttl 51, id 11799, len 40)


#12
05:23:15.253298 0:90:27:8c:e9:f3 0:10:67:0:85:7a ip 60:
192.168.1.10.http > 10.0.0.10.58702: . [tcp sum ok]
3225805158:3225805158(0) ack 4211792956 win 32120 (DF) (ttl 63, id
59370, len 40)
```

The victim server acknowledges the attacker's request in packet 5, and the server replies with a 404 (not found) in packet 6. The server even had time to issue a FIN packet before the NIDS injected a Reset (approx 20 mS after the "attack). Packets 10-12 indicate a normal session FIN. The forged packets are further identified as unique by the source MAC address `0:90:27:8b:17:17,` which is the hardware address for the Snort NIDS interface.

It is worth noting that the forged packets used the same sequence number (`4211792473`)as the attack. This assumes that the Operating System's TCP/IP stack will overwrite or replace any traffic received with the same starting sequence for the reset to be effective.

Sources:

1. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. (1998) – Ptacek and Newsham
2. Intrusion Signatures and Analysis (2001) - Northcutt, Cooper, Fearnow, Frederick. ISBN: 0-7357-1063-5
3. Know Your Enemy, Reveling the Security Tools, Tactics, and Motives of the Blackhat Community. (2001) – The Honeynet Project
ISBN: 0-201-74613-1
4. Network Intrusion Detection, An Analysts Handbook 2nd Edition (2001) - Northcutt and Novak.
ISBN: 0-7357-1008-2
5. Snort Users Manual: Snort Release 1.8.3 (2001) – Martin Roesch
6. TCP/IP Illustrated, Volume 1 (1994) - W. Richard Stevens
ISBN: 0-201-63346-9

Assignment 2 – Network Detects

Detect #1 – Hack-A-Tack Scan

a. Snort Data:

```
-----------------------------------------------------------------------
#(3 - 6573)[2001-09-21 04:16:38] IDS314/trojan_trojan-probe-hack-a-tack
IPv4: 212.70.42.126 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=29 ID=26383 flags=0 offset=0 TTL=41 chksum=2777
UDP:  port=31790 -> dport: 31789 len=9
Payload:  length = 1

000 : 41                                                    A

-----------------------------------------------------------------------
```

1. Source of Trace:

.org website that I run from my home office. The server is running RedHat Linux 7.1 with a custom install.

2. Detect was generated by:

a. Snort Intrusion Detection System v1.8.1.
b. Extracted from MySQL database via ACID v0.9.6b13.

The following rule was triggered:

```
alert UDP $EXTERNAL 31790 -> $INTERNAL 31789 (msg:
"IDS314/trojan_trojan-probe-hack-a-tack"; content: "A"; depth: 1;)
```

3. Probability the source address was spoofed:

The probability that this packet was spoofed is very low. The attacker was apparently looking for compromised machines by running the client side scanner, which provides the appropriate stimulus to stir a response from the server counterpart. The scanning system would need to receive an answer to identify compromised systems.

4. Description of attack:

Hack'a'Tack is a RAT (Remote Access Trojan) that allows an intruder to operate the victim's computer remotely without permission. It operates similar to well known RATs such as Back Orifice or SubSeven by allowing the attacker to view and interact with the victim's OS and applications. The server also allows the attacker to log keystrokes, reboot/halt the victim, and move files without the victim's permission.

5. Attack mechanism:

The client software, run by the attacker, includes a scanner that sends (in the Hack-A-Tack 2000 version) a single UDP packet to each IP at port 31789. A single character payload ("A") is used to further identify to the server that the Hack-A-Tack client is attempting a connection.

The victim must run the server side component to compromise the machine. This is typically done by fooling the victim into executing the installer by presenting it (by email, floppy, etc) as a non-malicious application. Tools are also available to package similar Trojans with other applications. Upon execution, the installer edits the victim's registry to load the Trojan during each reboot. The Trojan installs on Windows 9x and Windows NT4.0.

The client and Trojan server are available from the "authors" at http://www.rathat.de/

The following is Glocksoft's (http://www.glocksoft.com) summary of the RAT.

http://www.glocksoft.com/trojan_list/Hack_a_Tack.htm

Name:	Hack´a´Tack
Aliases:	HAT,
Ports:	31785, 31787, 31788, 31789 (UDP), 31790, 31791 (UDP), 31792
Files:	Hack'a'Tack.zip - 527,429 bytes Hack'a'tack110.zip - 537,799 bytes Hack'a'tack112.zip - 611,902 bytes Hacktack120.zip - 631,835 bytes Hat2k.zip - Hat2000.zip - 744,423 bytes Hack´a´Tack.exe - 300,248 bytes Hack´a´Tack.exe - 304,893 bytes Hack´a´Tack.exe - 308,716 bytes Hack´a´Tack.exe - 317,868 bytes Hack´a´Tack.exe - 429,744 bytes Server.exe - 241,397 bytes Server.exe - 246,331 bytes Server.exe - 279,418 bytes Server.exe - 620,544 bytes Server.exe - 642,560 bytes Expl32.exe - Cfgwiz32.exe - Win32ip.cfg - variable no of bytes
Created:	May 1999
Requires:	N/A
Actions:	Remote Access / Hidden IP-Scanner The trojan is able to decrypt cached passwords.
Versions:	1.0, 1.10, 1.12, 1.20, 1.2 te, 1.2 se, 2.1, 2000, 2000b,
Registers:	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
Notes:	Works on Windows 95 and 98. With somebugfiexes it will work on NT as well. Now there exists two beafed up versions called v.SE and v.TE with many bugfixes included. Version 2000 is shareware!
Country:	written in Germany ??
Program:	N/A

6. Correlations:

The following report retrieved from google.com's cache of the now defunct whitehats.com website.

IDS314/TROJAN-PROBE-HACK-A-TACK

Packet Traces

```
00/00-23:23:23 attacker:31790 -> target:31789
UDP TTL:117 TOS:0x0 ID:26394
Len: 1
41                                                A

(the above is a packet trace reconstructed from live data contributed
by Phil Wood)
```

Background

A variation of the "Hack'a'Tack" trojan has been seen in the field for which the probe is a single UDP packet containing the letter "A". The source port is 31790 and the destination is 31789.

Credits

Phil Wood: packet trace evidence.

Contributor

Phil Wood

7. Evidence of active targeting:

Due to the nature of the scanning tool, this was likely a blind scan of a large block of IP addresses.

At the time of detection, no other IP addresses on the /24 subnet were active or monitored by an IDS, but an associate was able to confirm that the same source IP had scanned an IP address on an adjoining /16 subnet within 5 minutes prior. This increases the likelihood that this detect was not from a targeted attack.

8. Severity:

(Critical + Lethal) – (System + Network Countermeasures) = Severity

(4+5)-(5+5) = -1

Critical – (4) This server provides web services for the .org website.
Lethal – (5) Detection of a compromised server would provide administrative control to the attacker.

System Countermeasures (5) The target of this attack runs an Operating System that is not capable of executing the server Trojan. (Red Hat Linux 7.1).
Network Countermeasures – (5) The server at this IP address is behind a firewall that only passes inbound TCP 80,

9. Defensive recommendation:

The defenses at this address are adequate for preventing this attack. The server component cannot execute on this operating system, which would render social engineering or other trickery useless. Because this network did not support end users, and (at the time) only supported a web server, the border firewall only passed TCP traffic inbound to port 80. All other inbound traffic was blocked at the time.

10. Multiple choice test question:

RATs are used by attackers to:

A. Identify the UDP ports listening on a potential victim.
B. Tunnel through firewalls by sending single byte payloads on ephemeral UDP ports.
C. Allow the attacker to control the victim machine remotely.
D. Initiate DNS zone transfers via UDP.

The correct answer is C.

Remote Access Trojans (RATs) allow the attacker to gain control of the victim machine. The victim user must be socially engineered or tricked into installing the Trojan service before the attacker can gain control.

a. Snort Data:

```
------------------------------------------------------------------------
#(1 - 332) [2001-11-15 23:19:28]  TCP SA inbound
IPv4: 65.203.157.138 -> xxx.xxx.xxx.xxx
      hlen=5 TOS=0 dlen=44 ID=3565 flags=0 offset=0 TTL=242
chksum=31321
TCP:  port=80 -> dport: 27374  flags=***A**S* seq=2673688066
      ack=1902687206 off=6 res=0 win=64240 urp=0 chksum=38129
      Options:
       #1 - MSS len=4 data=05B4
Payload: none
------------------------------------------------------------------------
#(1 - 333) [2001-11-15 23:19:28]  TCP RST outbound
IPv4: xxx.xxx.xxx.xxx -> 65.203.157.138
      hlen=5 TOS=0 dlen=40 ID=185 flags=0 offset=0 TTL=255 chksum=47761
TCP:  port=27374 -> dport: 80  flags=*****R** seq=1902687206
      ack=0 off=5 res=0 win=0 urp=0 chksum=34060
Payload: none
------------------------------------------------------------------------
#(1 - 334) [2001-11-15 23:19:31]  TCP SA inbound
IPv4: 65.203.157.138 -> xxx.xxx.xxx.xxx
      hlen=5 TOS=0 dlen=44 ID=3566 flags=0 offset=0 TTL=242
chksum=31320
TCP:  port=80 -> dport: 1243  flags=***A**S* seq=2673688066
      ack=1902687206 off=6 res=0 win=64240 urp=0 chksum=38129
      Options:
       #1 - MSS len=4 data=05B4
Payload: none
------------------------------------------------------------------------
#(1 - 335) [2001-11-15 23:19:31]  TCP RST outbound
IPv4: xxx.xxx.xxx.xxx -> 65.203.157.138
      hlen=5 TOS=0 dlen=40 ID=186 flags=0 offset=0 TTL=255 chksum=47760
TCP:  port=12345 -> dport: 80  flags=*****R** seq=1902687206
      ack=0 off=5 res=0 win=0 urp=0 chksum=34060
Payload: none
------------------------------------------------------------------------
```

1. Source of Trace:

This machine was an experimental honeypot. It was build with a default RedHat 6.2 install (client). The honeypot was placed behind a port forwarding Linux IPChains firewall.

2. Detect was generated by:

c. Snort Intrusion Detection System v1.8.1.
d. Extracted from MySQL database via ACID v0.9.6b13.
   - The Following "catch-all" rules were triggered:

```
alert UDP $EXTERNAL any -> $INTERNAL any (msg: "TCP SA inbound"; flags:
"SA+";)

alert UDP $INTERNAL any -> $EXTERNAL any (msg: "TCP RST outbound";
flags: "R+";)
```

3. Probability the source address was spoofed:

Because the attacker was querying both TCP ports 27374 and 1243, it is likely that this was an attempt to identify availability of a service on those ports. The attacker would be expecting a response, or lack thereof, to indicate if the port is open, closed, ignoring, or blocked. Therefore the attacker would likely be using a valid address to receive a response.

4. Description of attack:

These two packets indicate a scan for a SubSeven backdoor based on the destination ports. While SubSeven scans are very common on the Internet, this one gained attention because the intruder took measures to hide the scanning activity by crafting them appear as portions of normal traffic.

The TCP packets used in the scan used a source port of 80, and high ports (well known backdoor ports) as the destination. The SYN-ACK flagged packets could appear as a second part of a three-way handshake, allowing the packets to pass through stateless firewalls and evade simple IDS rules. The duplicate ack and sequence numbers on both packets points to the possibility that the packets were crafted.

5. Attack mechanism:

Unknown. There are tools available that can craft similar packets, but the specific tool used in this is attack cannot be identified. I do not believe that this was an automated tool, as the two packets were logged three seconds apart.

6. Correlations:

Win Miller captured a similar scan on a ZoneAlarm log. Found at:
http://www.sans.org/y2k/practical/Win_Miller_GCIA.doc

```
20010417, 21:17:50, 63.208.157.51, 80, 207.192.132.183, 2049, TCP [FWIN]
```

Win's probe used similar source/destination port trickery to evade detection, but was also detected because it was not part of a HTTP session.

7. Evidence of active targeting:

Because a "nearby" firewall capable of logging similar traffic did not detect similar scans, the possibility exists that these packets were directed to this specific host. At the date of this attack, two other machines with "nearby" addresses on the same /24 subnet

were online. Both were behind a port forwarding Linux IPChains firewall that allows, but syslogs, any activity on well-known backdoor Trojan ports. Review of the firewall logs does not indicate any presence of activity from this source, on this date, or with a source port of 80.

This source IP never returned to continue scanning or attempt any follow-ups.

8. Severity:

(Critical + Lethal) – (System + Network Countermeasures) = Severity

(1+5)-(3+1) = 2

Critical – (1) This honeypot was not used for anything "production".
Lethal – (5) Detection of a compromised server would provide administrative control to the attacker.
System Countermeasures (3) The target of this attack runs an Operating System that is not capable of executing the SubSeven. (Red Hat Linux 6.2). However, since the system did not have any patches applied- the countermeasures were lowered to 3.
Network Countermeasures – (1) The packets were not filtered and reached the destination system.

9. Defensive recommendation:

The weakest links on this machine are the default install and lack of patching. While this was intended, a "live" machine should not be exposed in this manner.

A stateful firewall would control access to the machine and block similar scans masquerading as legitimate traffic. The machine should also be patched and/or upgraded to reduce the risk of exposing vulnerable services to the Internet.

10. Multiple choice test question:

TCP traffic with a source port of 80 should only be seen:

A. When ephemeral destination ports are blocked at the Firewall.
B. In replies from an HTTP server.
C. When an HTTP server is operating on an alternate port.
D. Both A and B.

The correct answer is B.

Like most TCP services, HTTP (web) servers send data on the same port that they listen to. Since most HTTP servers operate on port 80, HTTP replies from these servers have a source port of 80.

a. Snort Data:

```
------------------------------------------------------------------------
#(1 - 731) [2001-11-17 11:23:32]  Custom__Grims Ping!
IPv4: 141.28.225.70 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=63 ID=19542 flags=0 offset=0 TTL=100
chksum=15056
TCP:  port=4672 -> dport: 21  flags=***AP*** seq=2231854002
      ack=1091630342 off=5 res=0 win=64076 urp=0 chksum=65277
Payload:  length = 23

000 : 50 41 53 53 20 46 67 70 75 73 65 72 40 68 6F 6D   PASS Fgpuser@hom
010 : 65 2E 63 6F 6D 0D 0A                              e.com..
------------------------------------------------------------------------
#(1 - 732) [2001-11-17 11:23:32]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=40 ID=537 flags=0 offset=0 TTL=64 chksum=43284
TCP:  port=21 -> dport: 4672  flags=***A**** seq=1091630342
      ack=2231854025 off=5 res=0 win=32120 urp=0 chksum=29068
Payload: none
------------------------------------------------------------------------
#(1 - 733) [2001-11-17 11:23:32]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=88 ID=538 flags=0 offset=0 TTL=64 chksum=43235
TCP:  port=21 -> dport: 4672  flags=***AP*** seq=1091630342
      ack=2231854025 off=5 res=0 win=32120 urp=0 chksum=50541
Payload:  length = 48

000 : 32 33 30 20 47 75 65 73 74 20 6C 6F 67 69 6E 20   230 Guest login
010 : 6F 6B 2C 20 61 63 63 65 73 73 20 72 65 73 74 72   ok, access restr
020 : 69 63 74 69 6F 6E 73 20 61 70 70 6C 79 2E 0D 0A   ictions apply...
------------------------------------------------------------------------
#(1 - 734) [2001-11-17 11:23:32]  (138)Unknown Sig Name
IPv4: 141.28.225.70 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=51 ID=19554 flags=0 offset=0 TTL=100
chksum=15056
TCP:  port=4672 -> dport: 21  flags=***AP*** seq=2231854025
      ack=1091630390 off=5 res=0 win=64028 urp=0 chksum=36685
Payload:  length = 11

000 : 43 57 44 20 2F 70 75 62 2F 0D 0A                  CWD /pub/..
------------------------------------------------------------------------
#(1 - 735) [2001-11-17 11:23:32]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=40 ID=539 flags=0 offset=0 TTL=64 chksum=43282
TCP:  port=21 -> dport: 4672  flags=***A**** seq=1091630390
      ack=2231854036 off=5 res=0 win=32120 urp=0 chksum=29009
Payload: none
------------------------------------------------------------------------
#(1 - 736) [2001-11-17 11:23:33]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=69 ID=540 flags=0 offset=0 TTL=64 chksum=43252
TCP:  port=21 -> dport: 4672  flags=***AP*** seq=1091630390
```

```
      ack=2231854036 off=5 res=0 win=32120 urp=0 chksum=37831
Payload:  length = 29

000 : 32 35 30 20 43 57 44 20 63 6F 6D 6D 61 6E 64 20   250 CWD command
010 : 73 75 63 63 65 73 73 66 75 6C 2E 0D 0A            successful...
-------------------------------------------------------------------------
#(1 - 737) [2001-11-17 11:23:33]  (138)Unknown Sig Name
IPv4: 141.28.225.70 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=59 ID=19558 flags=0 offset=0 TTL=100
chksum=15044
TCP:  port=4672 -> dport: 21  flags=***AP*** seq=2231854036
      ack=1091630419 off=5 res=0 win=63999 urp=0 chksum=49372
Payload:  length = 19

000 : 4D 4B 44 20 30 31 31 31 31 37 31 39 32 39 32 30   MKD 011117192920
010 : 70 0D 0A                                          p..
-------------------------------------------------------------------------
#(1 - 738) [2001-11-17 11:23:33]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=103 ID=541 flags=0 offset=0 TTL=64
chksum=43217
TCP:  port=21 -> dport: 4672  flags=***AP*** seq=1091630419
      ack=2231854055 off=5 res=0 win=32120 urp=0 chksum=32384
Payload:  length = 63

000 : 35 35 30 20 30 31 31 31 31 37 31 39 32 39 32 30   550 011117192920
010 : 70 3A 20 50 65 72 6D 69 73 73 69 6F 6E 20 64 65   p: Permission de
020 : 6E 69 65 64 20 6F 6E 20 73 65 72 76 65 72 2E 20   nied on server.
030 : 28 55 70 6C 6F 61 64 20 64 69 72 73 29 0D 0A      (Upload dirs)..
-------------------------------------------------------------------------
#(1 - 739) [2001-11-17 11:23:34]  (138)Unknown Sig Name
IPv4: 141.28.225.70 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=54 ID=19565 flags=0 offset=0 TTL=100
chksum=15042
TCP:  port=4672 -> dport: 21  flags=***AP*** seq=2231854055
      ack=1091630482 off=5 res=0 win=63936 urp=0 chksum=60310
Payload:  length = 14

000 : 43 57 44 20 2F 70 75 62 6C 69 63 2F 0D 0A         CWD /public/..
-------------------------------------------------------------------------
#(1 - 740) [2001-11-17 11:23:34]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=40 ID=542 flags=0 offset=0 TTL=64 chksum=43279
TCP:  port=21 -> dport: 4672  flags=***A**** seq=1091630482
      ack=2231854069 off=5 res=0 win=32120 urp=0 chksum=28884
Payload: none
-------------------------------------------------------------------------
#(1 - 741) [2001-11-17 11:23:34]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=82 ID=543 flags=0 offset=0 TTL=64 chksum=43236
TCP:  port=21 -> dport: 4672  flags=***AP*** seq=1091630482
      ack=2231854069 off=5 res=0 win=32120 urp=0 chksum=31502
Payload:  length = 42

000 : 35 35 30 20 2F 70 75 62 6C 69 63 2F 3A 20 4E 6F   550 /public/: No
010 : 20 73 75 63 68 20 66 69 6C 65 20 6F 72 20 64 69    such file or di
020 : 72 65 63 74 6F 72 79 2E 0D 0A                     rectory...
-------------------------------------------------------------------------
```

```
#(1 - 742) [2001-11-17 11:23:35]  (138)Unknown Sig Name
IPv4: 141.28.225.70 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=60 ID=19568 flags=0 offset=0 TTL=100
chksum=15033
TCP:  port=4672 -> dport: 21  flags=***AP*** seq=2231854069
      ack=1091630524 off=5 res=0 win=63894 urp=0 chksum=56643
Payload:  length = 20

000 : 43 57 44 20 2F 70 75 62 2F 69 6E 63 6F 6D 69 6E   CWD /pub/incomin
010 : 67 2F 0D 0A                                       g/..
-------------------------------------------------------------------------
#(1 - 743) [2001-11-17 11:23:35]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=88 ID=544 flags=0 offset=0 TTL=64 chksum=43229
TCP:  port=21 -> dport: 4672  flags=***AP*** seq=1091630524
      ack=2231854089 off=5 res=0 win=32120 urp=0 chksum=27787
Payload:  length = 48

000 : 35 35 30 20 2F 70 75 62 2F 69 6E 63 6F 6D 69 6E   550 /pub/incomin
010 : 67 2F 3A 20 4E 6F 20 73 75 63 68 20 66 69 6C 65   g/: No such file
020 : 20 6F 72 20 64 69 72 65 63 74 6F 72 79 2E 0D 0A    or directory...
-------------------------------------------------------------------------
#(1 - 744) [2001-11-17 11:23:36]  (138)Unknown Sig Name
IPv4: 141.28.225.70 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=56 ID=19585 flags=0 offset=0 TTL=100
chksum=15020
TCP:  port=4672 -> dport: 21  flags=***AP*** seq=2231854089
      ack=1091630572 off=5 res=0 win=63846 urp=0 chksum=33286
Payload:  length = 16

000 : 43 57 44 20 2F 69 6E 63 6F 6D 69 6E 67 2F 0D 0A   CWD /incoming/..
-------------------------------------------------------------------------
#(1 - 745) [2001-11-17 11:23:36]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=84 ID=545 flags=0 offset=0 TTL=64 chksum=43232
TCP:  port=21 -> dport: 4672  flags=***AP*** seq=1091630572
      ack=2231854105 off=5 res=0 win=32120 urp=0 chksum=4386
Payload:  length = 44

000 : 35 35 30 20 2F 69 6E 63 6F 6D 69 6E 67 2F 3A 20   550 /incoming/:
010 : 4E 6F 20 73 75 63 68 20 66 69 6C 65 20 6F 72 20   No such file or
020 : 64 69 72 65 63 74 6F 72 79 2E 0D 0A               directory...
-------------------------------------------------------------------------
#(1 - 746) [2001-11-17 11:23:37]  (138)Unknown Sig Name
IPv4: 141.28.225.70 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=56 ID=19592 flags=0 offset=0 TTL=100
chksum=15013
TCP:  port=4672 -> dport: 21  flags=***AP*** seq=2231854105
      ack=1091630616 off=5 res=0 win=63802 urp=0 chksum=27637
Payload:  length = 16

000 : 43 57 44 20 2F 5F 76 74 69 5F 70 76 74 2F 0D 0A   CWD /_vti_pvt/..
-------------------------------------------------------------------------
#(1 - 747) [2001-11-17 11:23:37]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=84 ID=546 flags=0 offset=0 TTL=64 chksum=43231
TCP:  port=21 -> dport: 4672  flags=***AP*** seq=1091630616
      ack=2231854121 off=5 res=0 win=32120 urp=0 chksum=64228
```

```
Payload:  length = 44

000 : 35 35 30 20 2F 5F 76 74 69 5F 70 76 74 2F 3A 20   550 /_vti_pvt/:
010 : 4E 6F 20 73 75 63 68 20 66 69 6C 65 20 6F 72 20   No such file or
020 : 64 69 72 65 63 74 6F 72 79 2E 0D 0A               directory...
-------------------------------------------------------------------------
#(1 - 748) [2001-11-17 11:23:38]  (138)Unknown Sig Name
IPv4: 141.28.225.70 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=47 ID=19603 flags=0 offset=0 TTL=100
chksum=15011
TCP:  port=4672 -> dport: 21  flags=***AP*** seq=2231854121
      ack=1091630660 off=5 res=0 win=63758 urp=0 chksum=13252
Payload:  length = 7

000 : 43 57 44 20 2F 0D 0A                              CWD /..
-------------------------------------------------------------------------
#(1 - 749) [2001-11-17 11:23:38]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=69 ID=547 flags=0 offset=0 TTL=64 chksum=43245
TCP:  port=21 -> dport: 4672  flags=***AP*** seq=1091630660
      ack=2231854128 off=5 res=0 win=32120 urp=0 chksum=37469
Payload:  length = 29

000 : 32 35 30 20 43 57 44 20 63 6F 6D 6D 61 6E 64 20   250 CWD command
010 : 73 75 63 63 65 73 73 66 75 6C 2E 0D 0A            successful...
-------------------------------------------------------------------------
#(1 - 750) [2001-11-17 11:23:38]  (138)Unknown Sig Name
IPv4: 141.28.225.70 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=59 ID=19613 flags=0 offset=0 TTL=100
chksum=14989
TCP:  port=4672 -> dport: 21  flags=***AP*** seq=2231854128
      ack=1091630689 off=5 res=0 win=63729 urp=0 chksum=49275
Payload:  length = 19

000 : 4D 4B 44 20 30 31 31 31 31 37 31 39 32 39 32 35   MKD 011117192925
010 : 70 0D 0A                                          p..
-------------------------------------------------------------------------
#(1 - 751) [2001-11-17 11:23:38]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=103 ID=548 flags=0 offset=0 TTL=64
chksum=43210
TCP:  port=21 -> dport: 4672  flags=***AP*** seq=1091630689
      ack=2231854147 off=5 res=0 win=32120 urp=0 chksum=32017
Payload:  length = 63

000 : 35 35 30 20 30 31 31 31 31 37 31 39 32 39 32 35   550 011117192925
010 : 70 3A 20 50 65 72 6D 69 73 73 69 6F 6E 20 64 65   p: Permission de
020 : 6E 69 65 64 20 6F 6E 20 73 65 72 76 65 72 2E 20   nied on server.
030 : 28 55 70 6C 6F 61 64 20 64 69 72 73 29 0D 0A      (Upload dirs)..
-------------------------------------------------------------------------
#(1 - 752) [2001-11-17 11:23:39]  (138)Unknown Sig Name
IPv4: 141.28.225.70 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=54 ID=19628 flags=0 offset=0 TTL=100
chksum=14979
TCP:  port=4672 -> dport: 21  flags=***AP*** seq=2231854147
      ack=1091630752 off=5 res=0 win=63666 urp=0 chksum=60467
Payload:  length = 14
```

```
000 : 43 57 44 20 2F 75 70 6C 6F 61 64 2F 0D 0A        CWD /upload/..
-----------------------------------------------------------------------
#(1 - 753) [2001-11-17 11:23:39]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=82 ID=549 flags=0 offset=0 TTL=64 chksum=43230
TCP:  port=21 -> dport: 4672  flags=***AP*** seq=1091630752
      ack=2231854161 off=5 res=0 win=32120 urp=0 chksum=31389
Payload:  length = 42

000 : 35 35 30 20 2F 75 70 6C 6F 61 64 2F 3A 20 4E 6F   550 /upload/: No
010 : 20 73 75 63 68 20 66 69 6C 65 20 6F 72 20 64 69    such file or di
020 : 72 65 63 74 6F 72 79 2E 0D 0A                     rectory...
-----------------------------------------------------------------------
#(1 - 754) [2001-11-17 11:23:40]  (138)Unknown Sig Name
IPv4: 141.28.225.70 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=40 ID=19631 flags=0 offset=0 TTL=100
chksum=14990
TCP:  port=4672 -> dport: 21  flags=***A***F seq=2231854161
      ack=1091630794 off=5 res=0 win=63624 urp=0 chksum=62510
Payload: none
-----------------------------------------------------------------------
#(1 - 755) [2001-11-17 11:23:40]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=40 ID=550 flags=0 offset=0 TTL=64 chksum=43271
TCP:  port=21 -> dport: 4672  flags=***A**** seq=1091630794
      ack=2231854162 off=5 res=0 win=32120 urp=0 chksum=28479
Payload: none
-----------------------------------------------------------------------
#(1 - 756) [2001-11-17 11:23:40]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=77 ID=551 flags=0 offset=0 TTL=64 chksum=43233
TCP:  port=21 -> dport: 4672  flags=***AP*** seq=1091630794
      ack=2231854162 off=5 res=0 win=32120 urp=0 chksum=32229
Payload:  length = 37

000 : 32 32 31 20 59 6F 75 20 63 6F 75 6C 64 20 61 74   221 You could at
010 : 20 6C 65 61 73 74 20 73 61 79 20 67 6F 6F 64 62    least say goodb
020 : 79 65 2E 0D 0A                                    ye...
-----------------------------------------------------------------------
#(1 - 757) [2001-11-17 11:23:40]  (138)Unknown Sig Name
IPv4: XXX.XXX.XXX.XXX -> 141.28.225.70
      hlen=5 TOS=16 dlen=40 ID=552 flags=0 offset=0 TTL=64 chksum=43269
TCP:  port=21 -> dport: 4672  flags=***A***F seq=1091630831
      ack=2231854162 off=5 res=0 win=32120 urp=0 chksum=28441
Payload: none
-----------------------------------------------------------------------
#(1 - 758) [2001-11-17 11:23:41]  (138)Unknown Sig Name
IPv4: 141.28.225.70 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=40 ID=19644 flags=0 offset=0 TTL=100
chksum=14977
TCP:  port=4672 -> dport: 21  flags=*****R** seq=2231854162
      ack=0 off=5 res=0 win=0 urp=0 chksum=9375
Payload: none
```

1. Source of Trace:

.org website that I run from my home office. The server is running RedHat Linux 7.1 with a custom install.

2. Detect was generated by:

e. Snort Intrusion Detection System v1.8.1.
f. Extracted from MySQL database via ACID v0.9.6b13.
A rule similar to the following was triggered:

```
alert tcp $EXTERNAL any -> $INTERNAL 21 (msg: "Custom__Grims Ping!";
content: "gpuser"; depth: 6; tag: session, 30, seconds, src;)
```

The "(138)Unknown Sig Name" labels indicate packets that were tagged (as a session) by the initial Custom__Grims Ping" rule.

3. Probability the source address was spoofed:

The probability that this packet was spoofed is very low. The scanning tool must establish an FTP session to test the victim site, which means that the attacking host must receive the responses from the target system. To perform these scans, the tool did establish an FTP session.

4. Description of attack:

Grim's Ping is an automated scanning tool that tests FTP sites for directories that are writeable by the anonymous user. It includes a port scanner, and automated FTP client, and log parser to report vulnerable hosts. Once an FTP server is located, the tool logs in as an anonymous user with the password Xgpuser@home.com (Where X is a random uppercase alphabetical character) The script then changes to and attempts to create directories using commands such as:

CWD /pub/
MKD 011117192920p (random name)
CWD /public/
CWD /pub/incoming/
CWD /incoming/
CWD /_vti_pvt/
CWD /
MKD 011117192925p (random name)
CWD /upload/

The author describes the tool's purpose:
This program was released in hopes that the general public would get hooked on scanning public sites and would help "spread the wealth."
….where "spreading the wealth" is often utilizing breached servers as warez sites.

5. Attack mechanism:

Grim's Ping default components support scanning for FTP services, checking for writeable directories, and generating a site report.
Grim's Ping includes the basic components required to detect and scan ftp servers for writeable directories. Additional components are available including:

Ping Companion -	used to identify remote OS
Ping Online -	Scans through CGI enabled web servers
Dir Check -	Logs all directory/file permissions
WinGet -	Scanning tool to find WinGate servers to hide attacker's FTP scanning

The tool is available at the developer's site: http://grimsping.cjb.net/

6. Correlations:

A Discussion thread on DShield can be found at: http://viper.dshield.org/pipermail/dshield/2001-October/001668.html

Laurie Zirkle posted the following portsentry logs at:
http://www.incidents.org/archives/intrusions/msg02377.html

```
Nov 05 00:20:18 hostl proftpd[22131] hostl (pD4B9F4AA.dip.t-dialin.net[212.185.244.170]):
FTP session opened.
Nov 05 00:20:19 hostl proftpd[22131] hostl (pD4B9F4AA.dip.t-dialin.net[212.185.244.170]):
ANON anonymous: Login successful.
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:19 -0500] "PASS guest@here.com"
230 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:20 -0500] "CWD /pub/" 250 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:20 -0500] "MKD .011105061909p"
550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:21 -0500] "CWD /pub/incoming/"
550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:21 -0500] "CWD /public/" 550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:22 -0500] "CWD /incoming/" 250
-
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:23 -0500] "CWD /_vti_pvt/" 550
-
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:23 -0500] "MKD .011105061911p"
550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:24 -0500] "CWD /" 250 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:24 -0500] "MKD .011105061913p"
550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:25 -0500] "CWD /upload/" 550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:25 -0500] "CWD /wwwroot/" 550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:26 -0500] "CWD /ftp/" 550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:26 -0500] "CWD /ftproot/" 550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:27 -0500] "CWD /.tmp/" 550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:27 -0500] "CWD /temp/" 550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:28 -0500] "CWD /tmp/" 550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:28 -0500] "CWD /~tmp/" 550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:29 -0500] "CWD /bak/" 550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:29 -0500] "CWD /ftp/incoming/"
550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:30 -0500] "CWD /anonymous/" 550
-
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:30 -0500] "CWD /ftp/upload/"
550 -
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:31 -0500] "CWD /_vti_log/" 550
-
pD4B9F4AA.dip.t-dialin.net UNKNOWN ftp [05/Nov/2001:00:20:31 -0500] "CWD /bin/" 550 -
Nov 05 00:20:32 hostl proftpd[22131] hostl (pD4B9F4AA.dip.t-dialin.net[212.185.244.170]):
FTP session closed.
```

Many of the mailing list discussions identify another common item in the scans- The fact that over 25% of these scans originate from the wannadoo.fr domain.

7. Evidence of active targeting:

Grim's Ping is designed to scan network segments, and there are no signs of non-automated activity. Therefore I have no reason to believe that this was a targeted scan.

As with any attack, the source IP may not necessarily indicate the attacker's true address, since compromised or mis-configured hosts can allow attackers to "bounce" their attacks. This is further supported by the fact that Grim's Ping was written to use open Wingate proxies to hide the attacker's address.

8. Severity:

(Critical + Lethal) – (System + Network Countermeasures) = Severity

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

Critical – (4) The ftp server runs on the site's web server. It is intended to allow registered users to edit web content and share large documents to anonymous FTP users. The FTP server supports the site but is not critical to its operation.
Lethal – (1) A successful attack would not jeopardize the integrity of the server or information. If a warez site were published, excessive storage and network utilization could be the result.
System Countermeasures (4) The server does allow anonymous logins. However, anonymous users are chroot'd and do not have any write permissions. Any service vulnerabilities are kept up to date and patched as appropriate.
Network Countermeasures – (2) Since this service is intended for public access and the attack tests service configurations- network countermeasures for this attack are not applicable. A 2 was given because the server is behind a firewall and monitored with a NIDS.

9. Defensive recommendation:

This machine was adequately protected from this attack. If anonymous access is not required, it should be removed to restrict access to authorized accounts. If the service is not required, it should be disabled and blocked at the firewall.

10. Multiple choice test question:

Grim's Ping is a tool used to:

A. Test web servers for directories that allow browsing.
B. Automate FTP transfers between warez sites.

C. Locate and test FTP servers for non-chroot'd environments.
D. Locate and test FTP servers for directories writeable by anonymous users.

The correct answer is D.

Grim's Ping scans for FTP servers and tests for directories that anonymous users can write to.

Detect #4 – FormMail

a. Snort Data:

```
--------------------------------------------------------------------------
#(2 - 1952) [2001-12-10 11:50:27] [Bugtraq/1187] [CVE/CVE-1999-0172]
[arachNIDS/226]  WEB-CGI formmail access
IPv4: 216.143.33.55 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=383 ID=33313 flags=0 offset=0 TTL=112 chksum=27757
TCP:  port=3543 -> dport: 80  flags=***AP*** seq=312700175
      ack=1611259037 off=5 res=0 win=8760 urp=0 chksum=16547
Payload:  length = 339

000 : 47 45 54 20 2F 63 67 69 2D 62 69 6E 2F 66 6F 72   GET /cgi-bin/for
010 : 6D 6D 61 69 6C 2E 70 6C 3F 72 65 63 69 70 69 65   mmail.pl?recipie
020 : 6E 74 3D 6B 6C 6A 66 64 73 38 39 39 34 79 32 72   nt=kljfds8994y2r
030 : 40 61 6F 6C 2E 63 6F 6D 26 73 75 62 6A 65 63 74   @aol.com&amp;subject
040 : 3D 56 69 6E 63 65 20 73 61 69 64 20 54 75 65 73   =Vince said Tues
050 : 64 61 79 26 65 6D 61 69 6C 3D 6B 6C 6A 64 73 66   day&amp;email=kljdsf
060 : 38 6A 32 33 66 6B 6A 73 6B 64 66 40 61 6F 6C 2E   8j23fkjskdf@aol.
070 : 63 6F 6D 26 3D 68 74 74 70 3A 2F 2F 77 77 77 2E   com&amp;=http://www.
080 : XX XX XX XX XX XX XX XX XX XX XX XX XX 2F 63      XXXXXXXXXXXXX/c
090 : 67 69 2D 62 69 6E 2F 66 6F 72 6D 6D 61 69 6C 2E   gi-bin/formmail.
0a0 : 70 6C 20 2E 70 6C 20 48 54 54 50 2F 31 2E 31 0D   pl .pl HTTP/1.1.
0b0 : 0A 41 63 63 65 70 74 3A 20 69 6D 61 67 65 2F 67   .Accept: image/g
0c0 : 69 66 2C 20 69 6D 61 67 65 2F 78 2D 78 62 69 74   if, image/x-xbit
0d0 : 6D 61 70 2C 20 69 6D 61 67 65 2F 6A 70 65 67 2C   map, image/jpeg,
0e0 : 20 69 6D 61 67 65 2F 70 6A 70 65 67 2C 20 2A 2F    image/pjpeg, */
0f0 : 2A 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 4D   *..User-Agent: M
100 : 69 63 72 6F 73 6F 66 74 20 55 52 4C 20 43 6F 6E   icrosoft URL Con
110 : 74 72 6F 6C 20 2D 20 36 2E 30 30 2E 38 38 36 32   trol - 6.00.8862
120 : 0D 0A 48 6F 73 74 3A 20 77 77 77 2E XX XX XX XX   ..Host: www.XXXX
130 : XX XX XX XX XX XX XX XX XX XX 0D 0A 43 61 63 68   XXXXXXXXXX..Cach
140 : 65 2D 43 6F 6E 74 72 6F 6C 3A 20 6E 6F 2D 63 61   e-Control: no-ca
150 : 63 68 65                                          che
```

b. Apache Log:

216.143.33.55 - - [10/Dec/2001:11:49:59 +0700] "GET /cgi-bin/formmail.pl?recipient=kljfds8994y2r@aol.com&subject=Vince%20said%20Tuesday&email=kljdsf8j23fkjskdf@aol.com&=http://www.XXXXXXXXXX.XXX/cgi-bin/formmail.pl HTTP/1.1" 404 300 "-" "Microsoft URL Control - 6.00.8862"

1. Source of Trace:

.org website that I run from my home office. The server is running RedHat Linux 7.1 with a custom install.

2. Detect was generated by:

g. Snort Intrusion Detection System v1.8.1.
h. Extracted from MySQL database via ACID v0.9.6b13.
The following rule was triggered:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI formmail
access";flags: A+; uricontent:"/formmail"; nocase;
reference:bugtraq,1187; reference:cve,CVE-1999-0172;
reference:arachnids,226; classtype:attempted-recon; sid:884; rev:2;)
```

3. Probability the source address was spoofed:

This source address was probably not spoofed. For the attacker to issue the HTTP request that tests the vulnerability, a TCP connection must be established.

4. Description of attack:

FormMail is a perl CGI program designed to parse HTTP form feedback and send an email to a recipient defined either by the site administrator or end user. The perl script is typically given the parameters via an HTTP POST.

This attack uses an HTTP GET to override any administrative settings, allowing the attacker to define the recipient as well as any form fields by including the strings in the HTTP request. Both crackers and spammers seek these relays because it allows them to send email anonymously by not including the attacker's IP address in the mail header. However, the HTTPd server logs will identify the attacker's IP address.

In this case, the GET request assumed the formmail.pl script was located in the default /cgi-bin/ directory. The request defines a recipient named "`kljfds8994y2r@aol.com`", and identifies the subject as "`Vince said Tuesday`". The attacker also forges sender's address as "`8j23fkjskdf@aol.com`"

The formmail software (including fixed releases) can be downloaded from the author at: http://www.worldwidemart.com/scripts/formmail.shtml

5. Attack mechanism:

Due to the increasing interest in locating mail relays, several tools have been published to exploit this vulnerability. Some sources are:
FormMail Scanner - http://www.geocities.com/dangerousonline2k1/myprogs.html
Formmail Bomber - http://www.angelfire.com/me3/wardaire2000/mailbombers.htm

The vulnerability can be easily executed by automated scan/attack tools or manually typed into a web browser to attack sites already known to be vulnerable.

6. Correlations:

Even the Incidents.org and Sans.org sites have detected scans. The following were posted by the Incidents.org handler on dust 7-27-2001.

Over the past few days we have received a number of attempts to relay mail using the sans.org and incidents.org servers.

```
--------------------
Jul 17 22:29:37 CGI-formmail: 24.129.74.155:3860 -> targetA:80
Jul 19 21:57:50 CGI-formmail: 141.158.63.16:4915 -> targetB:80
Jul 20 01:39:34 CGI-formmail: 209.244.208.132:65364 -> targetC:80
Jul 22 06:07:28 CGI-formmail: 172.139.111.116:2117 -> targetA:80
Jul 23 10:31:43 CGI-formmail: 172.186.199.178:4658 -> targetA:80
Jul 24 12:09:26 CGI-formmail: 65.29.53.232:1713 -> targetB:80
Jul 25 15:57:17 CGI-formmail: 141.157.94.243:3837 -> targetB:80
```

A decode of one of the CGI-formmail packets is below:

```
[**] IDS226 - CVE-1999-0172 - CGI-formmail [**]
07/25-15:57:17.161052 141.157.94.243:3837 -> targetB:80
TCP TTL:113 TOS:0x0 ID:38395 IpLen:20 DgmLen:469 DF
***AP*** Seq: 0x30B4787  Ack: 0x737534DF  Win: 0x2426  TcpLen: 20
47 45 54 20 2F 63 67 69 2D 62 69 6E 2F 66 6F 72  GET /cgi-bin/for
6D 6D 61 69 6C 2E 70 6C 3F 72 65 63 69 70 69 65  mmail.pl?recipie
6E 74 3D 65 6D 61 69 6C 40 61 64 64 72 65 73 73  nt=email@address
2E 63 6F 6D 2C 73 69 6C 6B 6B 39 64 34 40 61 6F  .com,silkk9d4@ao
6C 2E 63 6F 6D 26 73 75 62 6A 65 63 74 3D 68 74  l.com&subject=ht
74 70 3A 2F 2F 77 77 77 2E 73 61 6E 73 2E 6F 72  tp://www.sans.or
67 2F 63 67 69 2D 62 69 6E 2F 66 6F 72 6D 6D 61  g/cgi-bin/formma
69 6C 2E 70 6C 26 65 6D 61 69 6C 3D 50 6C 61 74  il.pl&email=Plat
69 6E 75 6D 53 63 61 6E 40 68 75 6E 74 65 72 2E  inumScan@hunter.
63 6F 6D 26 3D 68 74 74 70 3A 2F 2F 77 77 77 2E  com&=http://www.
73 61 6E 73 2E 6F 72 67 2F 63 67 69 2D 62 69 6E  sans.org/cgi-bin
```

2F 66 6F 72 6D 6D 61 69 6C 2E 70 6C 3C 62 72 3E /formmail.pl

An early article describing the attack can be found at:
http://securitytracker.com/alerts/2001/Mar/1001108.html

7. Evidence of active targeting:

This attempt, like most formmail scanning, is likely the result of an automated scan. The presence of a user agent in the GET string raised attention, but matched too many other scans to believe all attackers are manually entering similar attacks in the same browser versions.

8. Severity:

(Critical + Lethal) – (System + Network Countermeasures) = Severity

(4+3)-(5+1)=2

Critical – (4) This server provides web services for the .org website.
Lethal – (3) A vulnerable server can allow spammers to relay their traffic through the server. The direct effect would be bandwidth consumption, but could lead to the victim being added to the "Real-time Black hole List" (RBL). This would leave the victim unable to send mail to many recipients.
System Countermeasures – (5) The server does not have any CGI scripts available and replies to attacks with 404 "not found" errors.
Network Countermeasures – (1) Since the attack targets an application layer vulnerability, most network countermeasures are not effective in protecting the HTTPD services and underlying scripts.

9. Defensive recommendation:

Servers running FormMail should be running a current release (Version 1.9 or newer) to avoid this vulnerability. Any server running an older version should be upgraded or have the script removed immediately.

10. Multiple choice test question:

The FormMail script recipient, defined by the administrator, can be overridden by an attacker through:

A. flagging "cache control=nocache" in the form.
B. Using recipient= in the form POST.
C. Returning the form fields in a GET request, rather than a POST.
D. Flagging the packet with the form data as ACK_PSH.

The correct answer is C.

Returning the form data through a GET allows the attacker to override recipients defined by the administrator.

Detect #5 – Eazyspeed Trojan - Remote Scan Request

a. Snort Data:

```
------------------------------------------------------------------------
#(1 - 32047) [2001-09-29 05:43:03]  General TCP inbound
IPv4: 216.111.123.253 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=99 ID=1523 flags=0 offset=0 TTL=48 chksum=52997
TCP:  port=6667 -> dport: 1277  flags=***AP*** seq=2344042993
      ack=359983515 off=5 res=0 win=7300 urp=0 chksum=16967
Payload:  length = 59

000 : 3A 54 42 41 21 79 6F 64 61 40 66 20 50 52 49 56   :TBA!yoda@f PRIV
010 : 4D 53 47 20 23 75 6E 66 69 6E 67 20 3A 21 70 61   MSG #unfing :!pa
020 : 73 73 20 66 66 66 65 66 65 6D 66 65 65 33 65 69   ss fffefemfee3ei
030 : 74 33 69 66 66 68 33 66 68 0D 0A                  t3iffh3fh..
------------------------------------------------------------------------
#(1 - 32048) [2001-09-29 05:43:03]  General TCP outbound
IPv4: XXX.XXX.XXX.XXX -> 216.111.123.253
      hlen=5 TOS=0 dlen=70 ID=14148 flags=0 offset=0 TTL=128
chksum=19921
TCP:  port=1277 -> dport: 6667  flags=***AP*** seq=359983515
      ack=2344043052 off=5 res=0 win=8549 urp=0 chksum=21718
Payload:  length = 30

000 : 4E 4F 54 49 43 45 20 54 42 41 20 3A 50 61 73 73   NOTICE TBA :Pass
010 : 77 6F 72 64 20 41 63 63 65 70 74 65 64 0A         word Accepted.
------------------------------------------------------------------------
#(1 - 32049) [2001-09-29 05:43:03]  General TCP inbound
IPv4: 216.111.123.253 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=40 ID=1604 flags=0 offset=0 TTL=48 chksum=52975
TCP:  port=6667 -> dport: 1277  flags=***A**** seq=2344043052
      ack=359983545 off=5 res=0 win=7300 urp=0 chksum=12698
Payload: none
------------------------------------------------------------------------
#(1 - 32050) [2001-09-29 05:43:18]  General TCP inbound
IPv4: 216.111.123.253 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=474 ID=3396 flags=0 offset=0 TTL=48
chksum=50749
TCP:  port=6667 -> dport: 1277  flags=***AP*** seq=2344043052
      ack=359983545 off=5 res=0 win=7300 urp=0 chksum=54500
Payload:  length = 434

000 : 3A 54 42 41 21 79 6F 64 61 40 66 20 50 52 49 56   :TBA!yoda@f PRIV
010 : 4D 53 47 20 23 75 6E 66 69 6E 67 20 3A 21 2D 20   MSG #unfing :!-
020 : 2F 61 6C 69 61 73 20 6A 65 6A 65 20 7B 20 76 61   /alias jeje { va
030 : 72 20 25 78 20 3D 20 24 72 61 6E 64 28 31 2C 32   r %x = $rand(1,2
040 : 35 35 29 20 7C 20 76 61 72 20 25 79 20 3D 20 32   55) | var %y = 2
050 : 31 36 20 7C 20 73 65 74 20 25 73 63 61 6E 2E 73   16 | set %scan.s
060 : 74 61 72 74 31 20 25 79 20 7C 20 73 65 74 20 25   tart1 %y | set %
070 : 73 63 61 6E 2E 70 65 72 6D 31 20 25 79 20 7C 20   scan.perm1 %y |
080 : 73 65 74 20 25 73 63 61 6E 2E 65 6E 64 31 20 25   set %scan.end1 %
090 : 79 20 7C 20 73 65 74 20 25 73 63 61 6E 2E 73 74   y | set %scan.st
0a0 : 61 72 74 32 20 25 78 20 7C 20 73 65 74 20 25 73   art2 %x | set %s
0b0 : 63 61 6E 2E 70 65 72 6D 32 20 25 78 20 7C 20 73   can.perm2 %x | s
0c0 : 65 74 20 25 73 63 61 6E 2E 65 6E 64 32 20 25 78   et %scan.end2 %x
0d0 : 20 7C 20 73 65 74 20 25 73 63 61 6E 2E 73 74 61    | set %scan.sta
```

```
0e0 : 72 74 33 20 30 20 7C 20 73 65 74 20 25 73 63 61   rt3 0 | set %sca
0f0 : 6E 2E 70 65 72 6D 33 20 30 20 7C 20 73 65 74 20   n.perm3 0 | set
100 : 25 73 63 61 6E 2E 65 6E 64 33 20 32 35 35 20 7C   %scan.end3 255 |
110 : 20 73 65 74 20 25 73 63 61 6E 2E 73 74 61 72 74    set %scan.start
120 : 34 20 30 20 7C 20 73 65 74 20 25 73 63 61 6E 2E   4 0 | set %scan.
130 : 70 65 72 6D 34 20 30 20 7C 20 73 65 74 20 25 73   perm4 0 | set %s
140 : 63 61 6E 2E 65 6E 64 34 20 32 35 35 20 7C 20 73   can.end4 255 | s
150 : 65 74 20 25 73 63 61 6E 2E 70 6F 72 74 20 32 37   et %scan.port 27
160 : 33 37 34 2C 31 32 34 33 20 7C 20 73 65 74 20 25   374,1243 | set %
170 : 73 63 61 6E 2E 6E 69 63 6B 20 23 73 33 78 30 72   scan.nick #s3x0r
180 : 20 7C 20 73 65 74 20 25 73 63 61 6E 2E 64 65 6C    | set %scan.del
190 : 61 79 20 31 20 7C 20 74 69 6D 65 72 73 63 61 6E   ay 1 | timerscan
1a0 : 20 30 20 31 20 73 63 61 6E 63 68 65 63 6B 20 7D    0 1 scancheck }
1b0 : 0D 0A                                             ..
-------------------------------------------------------------------
#(1 - 32051) [2001-09-29 05:43:18]  General TCP outbound
IPv4: XXX.XXX.XXX.XXX -> 216.111.123.253
      hlen=5 TOS=0 dlen=479 ID=14404 flags=0 offset=0 TTL=128
chksum=19256
TCP:  port=1277 -> dport: 6667  flags=***AP*** seq=359983545
      ack=2344043486 off=5 res=0 win=8115 urp=0 chksum=44769
Payload:  length = 439

000 : 50 52 49 56 4D 53 47 20 23 75 6E 66 69 6E 67 20   PRIVMSG #unfing
010 : 3A 03 31 34 5B 03 31 32 64 6F 6E 65 03 31 34 5D   :.14[.12done.14]
020 : 1F 3A 1F 03 20 2F 2F 61 6C 69 61 73 20 6A 65 6A   .:.. //alias jej
030 : 65 20 7B 20 76 61 72 20 25 78 20 3D 20 24 72 61   e { var %x = $ra
040 : 6E 64 28 31 2C 32 35 35 29 20 7C 20 76 61 72 20   nd(1,255) | var
050 : 25 79 20 3D 20 32 31 36 20 7C 20 73 65 74 20 25   %y = 216 | set %
060 : 73 63 61 6E 2E 73 74 61 72 74 31 20 25 79 20 7C   scan.start1 %y |
070 : 20 73 65 74 20 25 73 63 61 6E 2E 70 65 72 6D 31    set %scan.perm1
080 : 20 25 79 20 7C 20 73 65 74 20 25 73 63 61 6E 2E    %y | set %scan.
090 : 65 6E 64 31 20 25 79 20 7C 20 73 65 74 20 25 73   end1 %y | set %s
0a0 : 63 61 6E 2E 73 74 61 72 74 32 20 25 78 20 7C 20   can.start2 %x |
0b0 : 73 65 74 20 25 73 63 61 6E 2E 70 65 72 6D 32 20   set %scan.perm2
0c0 : 25 78 20 7C 20 73 65 74 20 25 73 63 61 6E 2E 65   %x | set %scan.e
0d0 : 6E 64 32 20 25 78 20 7C 20 73 65 74 20 25 73 63   nd2 %x | set %sc
0e0 : 61 6E 2E 73 74 61 72 74 33 20 30 20 7C 20 73 65   an.start3 0 | se
0f0 : 74 20 25 73 63 61 6E 2E 70 65 72 6D 33 20 30 20   t %scan.perm3 0
100 : 7C 20 73 65 74 20 25 73 63 61 6E 2E 65 6E 64 33   | set %scan.end3
110 : 20 32 35 35 20 7C 20 73 65 74 20 25 73 63 61 6E    255 | set %scan
120 : 2E 73 74 61 72 74 34 20 30 20 7C 20 73 65 74 20   .start4 0 | set
130 : 25 73 63 61 6E 2E 70 65 72 6D 34 20 30 20 7C 20   %scan.perm4 0 |
140 : 73 65 74 20 25 73 63 61 6E 2E 65 6E 64 34 20 32   set %scan.end4 2
150 : 35 35 20 7C 20 73 65 74 20 25 73 63 61 6E 2E 70   55 | set %scan.p
160 : 6F 72 74 20 32 37 33 37 34 2C 31 32 34 33 20 7C   ort 27374,1243 |
170 : 20 73 65 74 20 25 73 63 61 6E 2E 6E 69 63 6B 20    set %scan.nick
180 : 23 73 33 78 30 72 20 7C 20 73 65 74 20 25 73 63   #s3x0r | set %sc
190 : 61 6E 2E 64 65 6C 61 79 20 31 20 7C 20 74 69 6D   an.delay 1 | tim
1a0 : 65 72 73 63 61 6E 20 30 20 31 20 73 63 61 6E 63   erscan 0 1 scanc
1b0 : 68 65 63 6B 20 7D 0A                              heck }.
-------------------------------------------------------------------
#(1 - 32052) [2001-09-29 05:43:19]  General TCP inbound
IPv4: 216.111.123.253 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=40 ID=3513 flags=0 offset=0 TTL=48 chksum=51066
TCP:  port=6667 -> dport: 1277  flags=***A**** seq=2344043486
      ack=359983984 off=5 res=0 win=7300 urp=0 chksum=11825
Payload: none
-------------------------------------------------------------------
#(1 - 32053) [2001-09-29 05:43:19]  General TCP inbound
IPv4: 216.111.123.253 -> XXX.XXX.XXX.XXX
```

```
      hlen=5 TOS=0 dlen=78 ID=3719 flags=0 offset=0 TTL=48 chksum=50822
TCP:  port=6667 -> dport: 1277  flags=***AP*** seq=2344043486
      ack=359983984 off=5 res=0 win=7300 urp=0 chksum=32658
Payload:  length = 38

000 : 3A 54 42 41 21 79 6F 64 61 40 66 20 50 52 49 56   :TBA!yoda@f PRIV
010 : 4D 53 47 20 23 75 6E 66 69 6E 67 20 3A 21 2D 20   MSG #unfing :!-
020 : 6A 65 6A 65 0D 0A                                 jeje..
-------------------------------------------------------------------------
#(1 - 32054) [2001-09-29 05:43:19]  General TCP outbound
IPv4: XXX.XXX.XXX.XXX -> 216.111.123.253
      hlen=5 TOS=0 dlen=83 ID=14660 flags=0 offset=0 TTL=128
chksum=19396
TCP:  port=1277 -> dport: 6667  flags=***AP*** seq=359983984
      ack=2344043524 off=5 res=0 win=8077 urp=0 chksum=23361
Payload:  length = 43

000 : 50 52 49 56 4D 53 47 20 23 75 6E 66 69 6E 67 20   PRIVMSG #unfing
010 : 3A 03 31 34 5B 03 31 32 64 6F 6E 65 03 31 34 5D   :.14[.12done.14]
020 : 1F 3A 1F 03 20 2F 6A 65 6A 65 0A                  .:.. /jeje.
-------------------------------------------------------------------------
#(1 - 32055) [2001-09-29 05:43:20]  General TCP inbound
IPv4: 216.111.123.253 -> XXX.XXX.XXX.XXX
      hlen=5 TOS=0 dlen=118 ID=3873 flags=0 offset=0 TTL=48
chksum=50628
TCP:  port=6667 -> dport: 1277  flags=***AP*** seq=2344043524
      ack=359984027 off=5 res=0 win=7300 urp=0 chksum=38816
Payload:  length = 78

000 : 3A 69 72 63 2E 75 6E 6C 65 65 74 2E 6E 65 74 20   :irc.unleet.net
010 : 34 30 34 20 6C 69 63 6B 6D 65 5B 35 34 37 30 5D   404 lickme[5470]
020 : 20 6C 69 63 6B 6D 65 5B 35 34 37 30 5D 20 3A 59    lickme[5470] :Y
030 : 6F 75 20 6E 65 65 64 20 76 6F 69 63 65 20 28 2B   ou need voice (+
040 : 76 29 20 28 23 75 6E 66 69 6E 67 29 0D 0A         v) (#unfing)..
-------------------------------------------------------------------------
#(1 - 32056) [2001-09-29 05:43:20]  General TCP outbound
IPv4: XXX.XXX.XXX.XXX -> 216.111.123.253
      hlen=5 TOS=0 dlen=40 ID=14916 flags=0 offset=0 TTL=128
chksum=19183
TCP:  port=1277 -> dport: 6667  flags=***A**** seq=359984027
      ack=2344043602 off=5 res=0 win=7999 urp=0 chksum=10967
Payload: none
-------------------------------------------------------------------------
```

1. Source of Trace:

The system with the obscured IP address (and using the IRC handle lickme[5470]) was intentionally infected with a Trojan "bot" to monitor the IRC channel being used to control infected Zombies suspected of performing mass scans for SubSeven Trojan backdoors.

2. Detect was generated by:

i. Snort Intrusion Detection System v1.8.1*.
j. Extracted from MySQL database via ACID v0.9.6b13.

*All packets logged (labeled by layer 4 protocol type and direction)

3. Probability the source address was spoofed:

This attack is executed through a previously established IRC session, which is connected via TCP. The IRC source address is expected to be the same as logged. However, the use of IRC hides the attacker's controls, since the commands are relayed through the IRC server. Therefore, the attacker's source IP is not logged here.

4. Description of attack:

The request to scan was logged roughly 50 hours after joining the IRC channel. The controller (identified as yoda@f) apparently authenticated with the zombie in packet (1 - 32047) by using the password "fffefemfee3eit3iffh3fh" sent via a private message.

Packet (1 – 32050) contained the following ASCII command:

```
#(1 - 32050)
:TBA!yoda@f PRIVMSG #unfing :!- /alias jeje { var %x = $rand(1,255) | var %y =
216 | set %scan.start1 %y | set %scan.perm1 %y | set %scan.end1 %y | set
%scan.start2 %x | set %scan.perm2 %x | set %scan.end2 %x| set %scan.start3 0 |
set %scan.perm3 0 | set %scan.end3 255 | set %scan.start4 0 | set %scan.perm4 0
| set %scan.end4 255 | set %scan.port 27374,1243 | set %scan.nick #s3x0r | set
%scan.delay 1 | timerscan 0 1 scancheck }..
```

The instructions identify the IP range to scan, including the first octet as 216 (scan.start/finish1), and the second as a single random number (using $RAND(1.255)). Both remaining octets have a range of 0-255, requesting a complete scan of the /16 netmasked range. Two destination ports (27374 and 1243) are requested through %scan.port. These are well known ports for SubSeven backdoors.

After the following command was issued, the zombie began scanning:

```
:irc.unleet.net 404 lickme[5470] lickme[5470] :You need voice (+v) (#unfing)..
```

The zombie began SYN scanning both ports at 216.122.0.0, incrementing the destination IPs. The port-forwarding IPChains firewall upstream of this host did not allow this traffic to pass, since it was configured to only pass IRC traffic outbound.

5. Attack mechanism:

The victim executed a binary program downloaded from http://home.dal.net/siner/setup.exe. It was known that several SubSeven backdoored hosts on the Internet were instructed to download and execute the same file after being scanned for the backdoor.

The "bot" contains the IRC client and scripting engine to carry out the attacker's requests. This bot was hard coded to check into this specific IRC server and channel. Many bots are available for download and modification; so a specific tool could not be identified by this example.

6. Correlations:

The Incidents.org handler on duty for September 25, 2001 posted information about SubSeven scanning and the binary files that the victims were instructed to download and execute. The article provided the location for the Trojan bot, which was used for this machine.

The article, Posted September 25, 2001, can be found at:
http://www.incidents.org/diary/september2001.php

```
=====================================================================
EasySpeed Trojans Still Posted, SubSeven Scans Still High
---------------------------------------------------------
Many sites are still receiving large numbers of scans to port
27374/tcp.
One class B military site is still receiving upwards of ten scans
per day from different souces. As expected, most attackers are
home user machines on cable, DSL, and dial-up connections.

The following trojans are currently hosted on home.dal.net. Recipients
of SubSeven (port 27374) probes are often instructed to fetch and
execute
these programs. Attempts to convince dal.net to remove the trojans have
as yet been unsuccessful.

http://home.dal.net/siner/setup.exe
http://home.dal.net/easyspeed/easyspeed.exe

=====================================================================
```

7. Evidence of active targeting:

This attack sequence targets the zombie victims that have checked into the IRC channel that the attacker has prepared or reserved for this purpose. These requests can only be issued to machines currently in the IRC channel, therefore the attacker (controller) is directly targeting groups of systems. In this case, the controller issued instructions using private messaging.

Given the use of these zombies to scan for more SubSeven backdoors, it is possible that most of these zombie systems are prior victims of completely random SubSeven backdoor scanning.

8. Severity:

(Critical + Lethal) – (System + Network Countermeasures) = Severity

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

Critical – (1) This system was installed to monitor the IRC channel and did not serve any other purpose. This number would be much higher for most compromised PCs.
Lethal – (4) The attacker has administrative control of the system. However, the attacker is likely to use the zombie for further attacks, leaving the system unscathed.
System Countermeasures – (1) The host has already been compromised, is running Trojan code, and publicized its presence on IRC. It was literally waiting for this attack.
Network Countermeasures – (2) Countermeasures to prevent the infection were not in place (as intended). Countermeasures were in place, however, to protect the Internet community from this host by restricting outbound traffic.

9. Defensive recommendation:

To defend against this attack, the system must be protected from precursor Trojan attacks. Many trojan attacks can be avoided by running anti-virus software and not downloading or executing untrusted software.

While the above cannot guarantee that zombie Trojans will not be installed, network countermeasures can assist in isolating infected victims from the IRC control channel. If IRC access is not necessary, a port filtering router/firewall can be used to block IRC traffic inbound or outbound.

10. Multiple choice test question:

The source address seen during mass distributed SubSeven scans will indicate:

A. The attacker's (controller's) IP address.
B. The IP address of the IRC server.
C. A common forged source address.
D. The address of the zombie hosts performing the scan.

The correct answer is D.

The zombie hosts perform the scans using their own IP as the source. The instructions to scan come from a communications channel, such as IRC, that the zombie systems listen to.

Assignment 3 – Analyze This

1. Introduction

We were asked to assist in the analysis of five days of NIDS logs to assist in identifying abuses of your network, assist in reducing unnecessary event counts, and to help automate log analysis. The data was collected by your Snort deployment from November 7 through 11, 2001. We have reviewed the top 20 alert and scan types with explanations of the threats and threats detected.

2. Log Files Analyzed

The following logs were analyzed:

alert.011107.gz	oos_Nov.7.2001.txt	scans.011107.gz
alert.011108.gz	oos_Nov.8.2001.txt	scans.011108.gz
alert.011109.gz	oos_Nov.9.2001.txt	scans.011109.gz
alert.011110.gz	oos_Nov.10.2001.txt	scans.011110.gz
alert.011111.gz	oos_Nov.11.2001.txt	scans.011111.gz

The logs contained a combined total of over 4.2 Million events. Upon initial review, I noticed that the scan logs contained a large percentage of DHCP chatter logged as scans initiated by the server- accounting for nearly 72% of the scans recorded during these dates. Approximately 2,960,000 records of DHCP traffic (originating from the DHCP server(s) EDU.NET.5.75 and EDU.NET.5.76) were eliminated from the scan logs before further analysis.

Nov 7 00:14:20 MY.NET.5.75:67 -> MY.NET.221.154:68 UDP
Nov 7 00:14:18 MY.NET.5.76:67 -> MY.NET.206.222:68 UDP

The breakdown by log type, after omission of DHCP chatter, includes the following:

alert.0111xx.gz	409,217 records
oos_Nov.xx.2001.txt	3,723 records
scans.0111xx.gz	832,374 records

3. Executive Summary

Our review of your Snort sensor data indicates that your sensor configurations log an excessive number of events. An average of 840,000 events were recorded per day during the time of our review. If your ID analysts are unable to review the records effectively, the likelihood that they would miss, or incorrectly analyze an event, increases. Your sensor configurations should be reviewed to reduce logging of repeating false positives and non-suspect traffic.

Communications that are non-priority or informational only are often logged. While traffic of this type, such as game and Internet file sharing, do not pose a security risk; your policies should define whether these should be addressed or removed from your sensor signatures to facilitate review.

We have identified several broad scans for service availability and for information gathering that should be looked into by your ID analysts and systems engineers. Some systems have been labeled as possible victims, and should be checked or removed from your network.

Most of your suspicious traffic appears to come from overseas addresses. Most of these are from China, yet are outside of the watchlists you have already defined for other networks in China. Because of this, we recommend creating additional watchlists to monitor netblocks that are identified as sources of undetermined yet suspicious traffic. Additionally, we have included recommendations to expand your logging depth on traffic types that can yield prudent information about the traffic traversing your network.

We have also included recommended steps to reduce the load on your ID analysts and limit common malicious traffic.

4. Detect List

Top 25 Event Types - Sorted by Number of events

Rank	Signature	# Alerts	# Sources	# Destinations
1	UDP scan	663708	139	17412
2	TCP ******S* scan	163435	212	22749
3	MISC Large UDP Packet	85455	30	55
4	MISC traceroute	43016	172	45
5	WEB-MISC prefix-get //	40049	1473	3
6	MISC source port 53 to <1024	38929	8289	10
7	INFO MSN IM Chat data	31744	351	523
8	CS WEBSERVER - external web traffic	26922	4314	1
9	SMB Name Wildcard	26639	669	9659
10	SCAN Proxy attempt	20203	250	11581
11	Incomplete Packet Fragments Discarded	20105	14	26
12	ICMP Echo Request BSDtype	16018	26	24
13	High port 65535 udp - possible Red Worm - traffic	8165	45	52
14	ICMP Fragment Reassembly Time Exceeded	6310	69	74
15	ICMP Destination Unreachable (Host Unreachable)	5484	370	62
16	Watchlist 000222 NET-NCFC	3271	18	12
17	SYN-FIN scan!	3068	1	3068
18	ICMP Echo Request Nmap or HPING2	3033	53	383
19	INFO Inbound GNUTella Connect accept	3032	42	2311
20	SMTP relaying denied	1777	7	20

5. Detect Descriptions and Correlations

5.1. UDP Scan

Signature	# Alerts	# Sources	# Destinations
UDP scan	663708	139	17412

These events were triggered by the Snort Portscan Preprocessor, which triggers upon detecting packets/datagrams sent to a specified number of ports over a specified period of time.

The default configuration:

preprocessor portscan: $HOME_NET 4 3

will trigger if one source IP sends TCP or UDP packets/datagrams to 4 or more ports over a period of 3 seconds.

The highest source of these events, EDU.NET.160.114 (401,841 events) is apparently running a Half-Life online game server. According to http://www.incidents.org/detect/gaming.php and http://clientbot.narod.ru/hl_anticheats.html UDP port 27005 is opened by Half-Life clients to receive traffic from the server. This machine was sending datagrams to the client ports of the recipients continuously throughout the 5 days. The fact that, at any given time, datagrams were being sent repeatedly to a group of recipients, further raised suspicion that this is a Half-Life server.

```
Nov 10 19:22:31 EDU.NET.160.114:888 -> 4.62.138.219:27005 UDP
Nov 10 19:22:31 EDU.NET.160.114:999 -> 24.218.53.216:27005 UDP
Nov 10 19:22:31 EDU.NET.160.114:888 -> 24.8.73.61:27005 UDP
Nov 10 19:22:31 EDU.NET.160.114:999 -> 204.155.149.59:27005 UDP
Nov 10 19:22:31 EDU.NET.160.114:999 -> 166.84.159.101:27005 UDP
Nov 10 19:22:31 EDU.NET.160.114:999 -> 142.166.219.42:27005 UDP
Nov 10 19:22:31 EDU.NET.160.114:888 -> 213.25.217.246:27005 UDP
Nov 10 19:22:31 EDU.NET.160.114:888 -> 172.139.242.68:27005 UDP
Nov 10 19:22:31 EDU.NET.160.114:888 -> 24.251.187.190:27005 UDP
Nov 10 19:22:31 EDU.NET.160.114:888 -> 24.13.27.232:27005 UDP
Nov 10 19:22:32 EDU.NET.160.114:888 -> 4.62.138.219:27005 UDP
```

This preprocessor is

Most of the remaining alerts are spread sparsely across hundreds of other sources, indicating the possibility that there are several false positive detects.

Legitimate sources, such as the following active mail server, can trigger this rule during the course of sending queries to DNS servers to identify senders and recipients.
Source IP: mailserver-ng.cs.YOUR.edu

```
Nov 10 04:27:15 EDU.NET.100.230:32781 -> 158.43.193.68:53 UDP
Nov 10 04:27:16 EDU.NET.100.230:32781 -> 136.187.17.2:53 UDP
Nov 10 04:27:19 EDU.NET.100.230:63651 -> 62.190.22.163:113 SYN ******S*
```

Nov 10 04:27:23 EDU.NET.100.230:32781 -> 24.14.77.14:53 UDP
Nov 10 04:27:23 EDU.NET.100.230:63653 -> 65.10.73.242:25 SYN ******S*
Nov 10 04:27:24 EDU.NET.100.230:32781 -> 136.187.17.2:53 UDP
Nov 10 04:27:29 EDU.NET.100.230:32781 -> 192.35.51.30:53 UDP
Nov 10 04:27:33 EDU.NET.100.230:32781 -> 136.187.17.2:53 UDP
Nov 10 04:27:34 EDU.NET.100.230:32781 -> 205.158.184.102:53 UDP
Nov 10 04:27:34 EDU.NET.100.230:63655 -> 12.152.164.199:113 SYN ******S*
Nov 10 04:27:34 EDU.NET.100.230:32781 -> 63.88.172.10:53 UDP
Nov 10 04:27:35 EDU.NET.100.230:32781 -> 209.185.130.68:53 UDP
Nov 10 04:27:39 EDU.NET.100.230:32781 -> 136.187.17.2:53 UDP
Nov 10 04:27:43 EDU.NET.100.230:32781 -> 136.187.17.2:53 UDP
Nov 10 04:41:10 EDU.NET.100.230:32781 -> 211.47.45.22:53 UDP
Nov 10 04:41:11 EDU.NET.100.230:63877 -> 159.149.70.90:25 SYN ******S*
Nov 10 04:41:11 EDU.NET.100.230:32781 -> 140.112.30.21:53 UDP
Nov 10 04:41:11 EDU.NET.100.230:32781 -> 204.91.99.140:53 UDP

5.2. TCP Syn Scan

Signature	# Alerts	# Sources	# Destinations
TCP ******S* scan	163435	212	22749

Like the UDP scans, these events were triggered by the Snort Portscan Preprocessor. The Portscan Preprocessor triggers upon detecting packets sent to a specified number of ports over a specified period of time.

The default configuration:

preprocessor portscan: $HOME_NET 4 3

will trigger if one source IP sends TCP or UDP packets/datagrams to 4 or more ports over a period of 3 seconds.

A majority of these events appear to be false positives triggered by machines that are connecting to several systems and/or services within short periods of time. Many of the

sources with high event counts triggered when sending SYNs to multiple destination IP's. These were commonly triggered by servers such as:

- EDU.NET.5.76 jupiter.noc.YOUR.edu multiple connects to TCP 23 (Telnet)
- EDU.NET.100.230 mailserver-ng.cs.YOUR.edu multiple connects to TCP 25 (SMTP)
- EDU.NET.253.24 listproc.YOUR.edu multiple connects to TCP 25 (SMTP)

A few legitimate scans were detected, but were limited to small numbers of hosts and had few or no follow up attempts.

One example of a true scan detect is the following "noisy" scan originating from 63.150.23.120 (centipede.symmetric.net) that sent 1363 SYN requests to EDU.NET.158.102 (does not resolve in DNS) to various destination ports, in random order.

```
Nov 10 09:59:03 63.150.23.120:2175 -> EDU.NET.158.102:1534 SYN ******S*
Nov 10 09:59:03 63.150.23.120:2176 -> EDU.NET.158.102:129 SYN ******S*
Nov 10 09:59:03 63.150.23.120:2177 -> EDU.NET.158.102:1549 SYN ******S*
Nov 10 09:59:03 63.150.23.120:2178 -> EDU.NET.158.102:3457 SYN ******S*
Nov 10 09:59:03 63.150.23.120:2179 -> EDU.NET.158.102:538 SYN ******S*
Nov 10 09:59:03 63.150.23.120:2181 -> EDU.NET.158.102:1453 SYN ******S*
Nov 10 09:59:03 63.150.23.120:2182 -> EDU.NET.158.102:1458 SYN ******S*
Nov 10 09:59:04 63.150.23.120:2185 -> EDU.NET.158.102:5001 SYN ******S*
Nov 10 09:59:04 63.150.23.120:2186 -> EDU.NET.158.102:5632 SYN ******S*
Nov 10 09:59:04 63.150.23.120:2187 -> EDU.NET.158.102:2023 SYN ******S*
Nov 10 09:59:04 63.150.23.120:2188 -> EDU.NET.158.102:1354 SYN ******S*
Nov 10 09:59:04 63.150.23.120:2189 -> EDU.NET.158.102:94 SYN ******S*
Nov 10 09:59:04 63.150.23.120:2190 -> EDU.NET.158.102:195 SYN ******S*
Nov 10 09:59:04 63.150.23.120:2191 -> EDU.NET.158.102:507 SYN ******S*
Nov 10 09:59:04 63.150.23.120:2192 -> EDU.NET.158.102:1248 SYN ******S*
Nov 10 09:59:04 63.150.23.120:2193 -> EDU.NET.158.102:1351 SYN ******S*
```

```
Nov 10 09:59:04 63.150.23.120:2194 -> EDU.NET.158.102:2232 SYN ******S*
Nov 10 09:59:04 63.150.23.120:2195 -> EDU.NET.158.102:7006 SYN ******S*
Nov 10 09:59:04 63.150.23.120:2196 -> EDU.NET.158.102:1669 SYN ******S*
Nov 10 09:59:04 63.150.23.120:2197 -> EDU.NET.158.102:362 SYN ******S*
Nov 10 09:59:04 63.150.23.120:2198 -> EDU.NET.158.102:1495 SYN ******S*
Nov 10 09:59:04 63.150.23.120:2199 -> EDU.NET.158.102:416 SYN ******S*
```

This source IP later triggered the following events:

- 2 instances of *SUNRPC highport access!*
- 2 instances of *TFTP - External TCP connection to internal tftp server*
- 3 instances of *INFO - Possible Squid Scan*
- 4 instances of *SCAN Proxy attempt*

5.3. Misc Large UDP Packet

Signature	# Alerts	# Sources	# Destinations
MISC Large UDP Packet	85455	30	55

Interestingly enough, one of the least descriptive signatures has raised the most interest. This signature triggers on UDP datagrams with a total payload larger than 4000 bytes. UDP traffic with payloads of this size, are not commonly seen on the Internet and may indicate file transfers, covert channels, or game traffic.

The signature that triggered these events is:
alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC Large UDP Packet"; dsize: >4000; reference:arachnids,247; classtype:bad-unknown; sid:521; rev:1;)

These events were logged simultaneously with several of the Incomplete Packet Fragments Discarded events discussed in section 5.11, presumably because the Snort sensors did not receive all fragments.

The top ten sources and destinations are shown in the following two tables:
Top Sources

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
61.134.9.88	26578	32880	22	22
61.150.5.18	21005	22059	7	7
61.150.5.19	13917	26014	9	9
61.175.133.20	7944	7945	1	1

61.153.17.24	6115	6115	1	1
61.153.116.195	3949	3949	1	1
210.124.186.227	1815	1817	2	2
213.244.175.42	1383	1385	3	3
211.40.179.122	1061	1665	3	3
211.233.58.22	396	396	1	1

Top Destinations

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
EDU.NET.111.221	38602	38609	4	9
EDU.NET.84.195	9215	9217	1	3
EDU.NET.53.40	5084	7138	1	13
EDU.NET.53.45	4247	4893	2	25
EDU.NET.53.49	3247	3348	1	13
EDU.NET.53.42	2478	2550	1	10
EDU.NET.53.32	2351	5535	1	13
EDU.NET.153.203	1766	1769	3	3
EDU.NET.153.144	1683	1698	2	7
EDU.NET.152.180	1644	1656	1	5

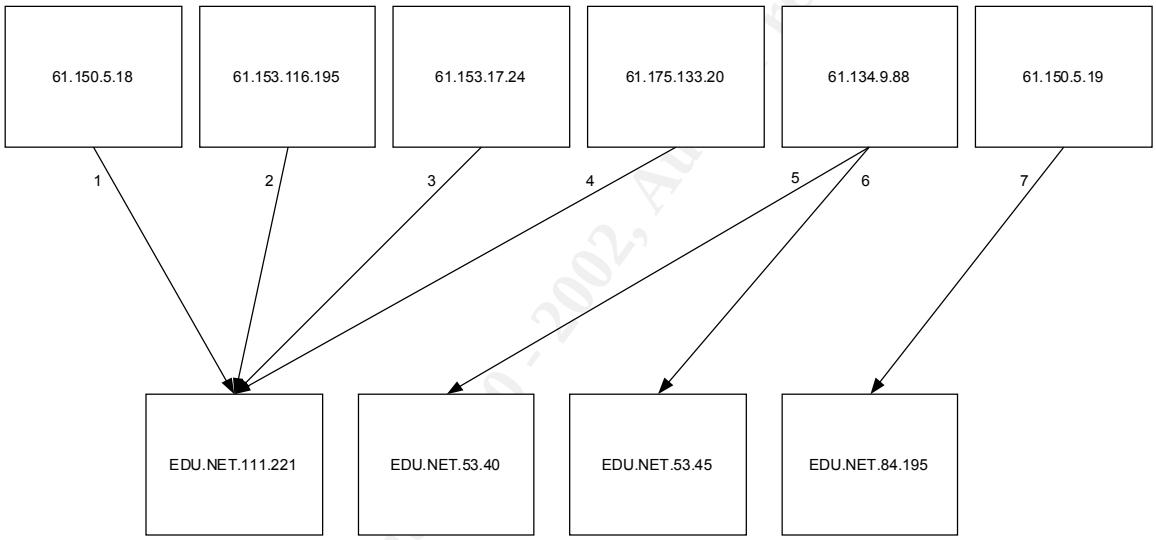
EDU.NET.111.221 (which does not resolve in DNS) is the only host that received similar traffic from more than 2 sources. The source 61.134.9.88 triggered alerts for similar traffic to 22 destinations, exceeding 61.150.5.19's 9 destinations.

Most of the source IP's are registered to overseas entities. The top 10 sources are resolved below.

Rank	IP	Entity (country)	Registry
1	61.134.9.88	XI'AN DATA BUREAU (China)	APNIC
2	61.150.5.18	xi'an data branch,XIAN CITY SHAANXI PROVINCE (China)	APNIC
3	61.150.5.19	xi'an data branch,XIAN CITY SHAANXI PROVINCE (China)	APNIC
4	61.175.133.20	CHINANET Zhejiang province network (China)	APNIC

5	61.153.17.24	Zhejiang Nation Tax Bureau,Hangzho (China)	APNIC
6	61.153.116.195	Zhejiang Nation Tax Bureau,Hangzhou (China)	APNIC
7	210.124.186.227	SPORTSNET2 (Korea)	KNIC
8	213.244.175.42	Level 3 Communications International (UK)	RIPE
9	211.40.179.122	DACOM-KIDC (Korea)	KNIC
10	211.233.58.22	KIDC-INFRA-SERVERHOSTING-INEMPIRE (Korea)	KNIC

Source/Destination Relationship
The following graph depicts the relationship between traffic from the top 6 sources and top 4 destinations.



Most traffic appears for periods of one to two hours, possibly indicating sessions of traffic. Since we do not have a duplicate Snort rule to correlate, we cannot be sure if this traffic is only flowing inbound, if the outbound traffic does not match any rules, or if the triggered rule only alerts on inbound traffic. It should also be noted that most of this traffic never exceed 5 packets per second, far too few to indicate a denial of service.

"Sessions" sorted by src/dest pairs in link graph

Src/Dest Line	Date	Start Time	End Time
1	Nov 7	10:59	13:55
	Nov 8	21:57	23:35
	Nov 9	9:03	10:15
	Nov 9	23:04	00:34 (Nov 10)
2	Nov 8	10:43	11:20
	Nov 9	9:03	9:08
	Nov 9	14:07	14:54

3	Nov 10	10:59	13:55
	Nov 11	16:58	18:03
4	Nov 7	15:50	17:08
	Nov 8	16:31	18:32
5	Nov 10	15:43	17:42
6	Nov 9	23:01	23:01
	Nov 10	15:17	17:14
	Nov 10	19:01	20:37
	Nov 10	22:46	23:22
7	Nov 9	18:21	18:32
	Nov 9	23:25	23:30
	Nov 10	11:26	14:35

Highlighted times indicate sessions that started within 5 minutes of another.

"Sessions" sorted by time

Src/Dest Line	Date	Start Time	End Time
1	Nov 7	10:59	13:55
4	Nov 7	15:50	17:08
2	Nov 8	10:43	11:20
4	Nov 8	16:31	18:32
1	Nov 8	21:57	23:35
2	Nov 9	9:03*	9:08
1	Nov 9	9:03	10:15
2	Nov 9	14:07	14:54
7	Nov 9	18:21	18:32
6	Nov 9	23:01	23:01
1	Nov 9	23:04	00:34 (Nov 10)
7	Nov 9	23:25	23:30
1	Nov 10	10:59	13:55
7	Nov 10	11:26	14:35
6	Nov 10	15:17**	17:14
5	Nov 10	15:43	17:42
6	Nov 10	19:01	20:37
6	Nov 10	22:46	23:22
3	Nov 11	16:58	18:03

*Times highlighted in red indicate overlapping sessions on a destination host
**Times highlighted in blue indicate overlapping sessions from a source host

We cannot rule out the possibility that this is harmless game traffic, but the geographic separation of sources does not support the likelihood. With the information available, this traffic appears suspicious, and we would recommend that this be looked into further. We recommend capturing payloads to assist in identifying the purpose of this traffic.

5.4. MISC Traceroute

Signature	# Alerts	# Sources	# Destinations
MISC traceroute	43016	172	45

Since the rule for this was not available for comparison (It is not in the current snort rule release using the same description.), a reasonable estimation must be made about the signature and the traffic that triggered the events. Without the availability of the Snort signature that was triggered, there is no way to confirm if the event triggers on the TTL=1 packet, or the ICMP TTL Expired reply. Therefore the source in the event could be the target (if the event is triggered by the ICMP reply).

Traceroute is a tool that maps a route to a destination by sending packets with incrementing TTLs. As each TTL expires, ICMP error messages are sent back by the device that expired the TTL, identifying the next "hop". Several layer 4 protocols can be used for tracing routes, as long as the IP TTL increments.

Traceroute, by itself, is not an attack. However, it can be used as a discovery tool to map networks, check availability of a device, or evade intrusion detection.

Because the triggered rule is not available for review, we must make an assumption as to the direction of traffic that triggered the alert. One destination (EDU.NET.140.9 – amp.noc.YOUR.edu) "received" a majority of the traceroutes (41,876), all from 57 "sources"- with each source probing this host only. If interest are the facts that each of the 57 "sources" tracerouted the "destination" 600-800 times, and that nearly all "sources" resolve in DNS to .edu domains. Given this information, it is likely that the events were triggered by the ICMP TTL Expired message, not by detection of a TTL of 1. Therefore, it is our belief that the host in your NOC has tracerouted the 57 sources (The top and bottom five are displayed to show the similar event counts).

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
128.197.160.253	783	783	1	1
141.142.121.7	781	781	1	1
128.182.61.50	771	771	1	1
141.219.100.16	767	767	1	1
128.3.7.27	765	765	1	1
---	---	---	---	---
160.36.56.49	709	709	1	1
140.180.128.45	706	706	1	1
205.253.57.100	689	689	1	1

134.129.107.72	688	688	1	1
199.249.169.82	680	680	1	1

5.5. WEB-MISC prefix-get //

Signature	# Alerts	# Sources	# Destinations
WEB-MISC prefix-get //	40049	1473	3

An HTTP GET request with a leading double slash can identify an attempt to evade simple ID systems, bypass authentication controls, or simply bad typing. A typical GET request will start with one leading slash to identify the web root.

The following signature was triggered.
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-MISC prefix-get //";flags: A+; uricontent:"get //"; nocase; classtype:attempted-recon; sid:1114; rev:2;)

The following three servers are the only destinations recorded by this signature:

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
EDU.NET.253.114	39439	39958	1414	1445
EDU.NET.253.115	608	653	79	84
EDU.NET.100.165	2	27651	2	4368

The CSWebserver received 98.5% of the logged events (39,439 of 40,049). The fact that this host received such a large percentage of requests points to targeted traffic. There are numerous external sources, and none stand out in quantity of requests or timing. Since the payload is not available, more detailed logging could identify the target directory/file and any possible bypass of authentication by forcing a different root directory.

5.6. Misc Source Port 53 to <1024

Signature	# Alerts	# Sources	# Destinations
MISC source port 53 to <1024	38929	8289	10

This indicates that a low numbered source port (53) was used to connect with a privileged (low - below 1024) port on your network. Normally, traffic connecting inbound to your network would originate from an ephemeral port (above 1023). False positives are commonly logged when DNS servers resolve addresses via TCP. Most of these will be seen with a source and destination port of 53. If malicious, this type of traffic may indicate an evasive technique that attempts to appear as a TCP DNS query originating from within your network.

These events are logged by the following rule:

alert tcp $EXTERNAL_NET 53 -> $HOME_NET :1023 (msg:"MISC source port 53 to <1024"; flags:S; reference:arachnids,07; classtype:bad-unknown; sid:504; rev:2;)

All recorded traffic had a source port of 53 and destination port of 53. This traffic typically indicates DNS resolutions larger than 512 bytes, and can indicate zone transfers. Since this rule only logs inbound traffic and does not log TCP flags, the querying source cannot identified. Since over 8,000 sources were identified by the logging, this does not appear to show signs of targeting. Therefore we do not believe any of this traffic to be malicious.

5.7. INFO MSN IM Chat Data

Signature	# Alerts	# Sources	# Destinations
INFO MSN IM Chat data	31744	351	523

These logged events indicate that hosts within your network are sending Instant Messaging (IM) chat messages outside your network. This Snort rule is for information only, as logged events do not indicate any attack. IM chat events are typically logged where policy dictates that chat messaging is not appropriate, or situations where sensitive information may be transferred through network perimeters via nonstandard means (web, email, etc).

The rule triggered is:
alert tcp $HOME_NET any <> $EXTERNAL_NET 1863 (msg:"INFO MSN IM Chat data";flags: A+; content:"|746578742F706C61696E|"; depth:100; classtype:not-suspicious; sid:540; rev:1;)

*Snort rules available for download during the past several months defaulted to triggering only on outbound traffic. We believe that your IDS rules administrator changed the direction to capture inbound and outbound traffic, since outside sources and inside destination s are captured in the logged events.

If policy permits IM traffic, these events offer little information other than identifying hosts that are engaged in IM chats. If policies restrict sensitive data transfers or require further scrutiny, full logging will capture the messaging payload for detailed review.

Given the assumption that IM messaging is permitted, the events merely indicate hosts with the most activity.

5.8. CS Webserver – External Web Traffic

Signature	# Alerts	# Sources	# Destinations
CS WEBSERVER - external web traffic	26922	4314	1

Our understanding is that this is a watchlist entry to monitor web access to the CS Webserver from outside your network. We do not have a copy of the custom rule, but we are assuming that the rule triggers on any external host/port handshaking with EDU.NET.100.65 port 80.

While the rule is believed to log only for correlation with other events, very few of the visiting hosts triggered other rules. The following were other alerts from the top 30 hosts.

- 45 instances of WEB-MISC http directory traversal
- 5 instances of WEB-CGI ksh access
- 3 instances of SCAN Proxy attempt
- 2 instances of WEB-MISC http directory traversal
- 1 instances of WEB-MISC handler access
- 1 instances of WEB-CGI ksh access
- 1 instances of WEB-CGI redirect access
- 1 instances of WEB-CGI archie access
- 1 instances of WEB-CGI formmail access
- 1 instances of WEB-CGI csh access

Excluding these few exceptions, this traffic does not appear suspect. It is worth noting that some hosts requested traffic at ~1 minute intervals during their visit. This may indicate that one or more pages on the CS Webserver auto-refresh, such a web cam, real-time status, or logging. If further information is required, more specific Snort rules should be applied.

5.9. SMB Name Wildcard

Signature	# Alerts	# Sources	# Destinations
SMB Name Wildcard	26639	669	9659

These events are the result of a SMB Name Wildcard query, which is a UDP request asking for names associated with a Windows machine or SAMBA server (Unix equivalent). The wildcard string requests that all network registered names be returned to the querying host. Registered names will include NetBIOS hostnames, usernames, workgroup/domain membership, and various network services such as the Server Service and Internet Information Server (IIS).

The following rule alerts on the SMB Name Wildcard:

alert UDP $EXTERNAL any -> $INTERNAL 137 (msg: "IDS177/netbios_netbios-name-query"; content: "CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA|00 00|";)

The nbtstat command/reply will appear similar to:

>nbtstat -A 192.168.1.1

NetBIOS Remote Machine Name Table

Name Type Status

```
----------------------------------------
MACHINE2    <00>  UNIQUE     Registered
MACHINE2    <20>  UNIQUE     Registered
DOMAIN3     <00>  GROUP      Registered
MACHINE2    <03>  UNIQUE     Registered
USER1       <03>  UNIQUE     Registered

MAC Address = 00-A0-C9-1F-94-6F
```

This example allows an attacker to resolve the following based on the "Name" and "Type":

- The target machine has a NetBIOS name of "MACHINE2"
- The Workstation service is started <00>
- The machine is a member of workgroup/domain "DOMAIN3"
- The File Server Service is started <20>
- The Messenger Service is started and accepts messages for "MACHINE2" and "USER1" <03>

A request can be generated from an attempt to connect to a SMB fileserver (by excluding the sharename), from a command line using nbtstat, or a scanning tool such as NetBIOS Name Network Scanner available at http://www.inetcat.org/software/nbtscan.html. Because false alerts can originate from legitimate file sharing use, queries with a source or destination outside of your network attract more attention.

Only one source from outside your network exceeded 33 queries over the period of five days. This host (216.150.152.145 - wiredforlife5.spyral.net) requested 3,807 SMB Name Wildcard requests and 3,807 L3Retriever Ping requests directed to a single host within your network (EDU.NET.5.44 tsunami.YOUR.edu).

The following events show the frequency of the queries.

```
11/08-04:30:37.407889 [**] SMB Name Wildcard [**] 216.150.152.145:137 ->
EDU.NET.5.44:137

11/08-04:31:17.269173 [**] ICMP Echo Request L3retriever Ping [**] 216.150.152.145
-> EDU.NET.5.44

11/08-04:31:17.277950 [**] SMB Name Wildcard [**] 216.150.152.145:137 ->
EDU.NET.5.44:137

11/08-04:31:36.113813 [**] ICMP Echo Request L3retriever Ping [**] 216.150.152.145
-> EDU.NET.5.44

11/08-04:31:52.619936 [**] ICMP Echo Request L3retriever Ping [**] 216.150.152.145
-> EDU.NET.5.44
```

```
11/08-04:31:52.633134 [**] SMB Name Wildcard [**] 216.150.152.145:137 ->
EDU.NET.5.44:137
11/08-04:31:54.919725 [**] SMB Name Wildcard [**] 216.150.152.145:137 ->
EDU.NET.5.44:137
11/08-04:32:11.489046 [**] ICMP Echo Request L3retriever Ping [**] 216.150.152.145
-> EDU.NET.5.44
11/08-04:32:11.502098 [**] SMB Name Wildcard [**] 216.150.152.145:137 ->
EDU.NET.5.44:137
11/08-04:32:13.742678 [**] ICMP Echo Request L3retriever Ping [**] 216.150.152.145
-> EDU.NET.5.44
11/08-04:32:13.759376 [**] SMB Name Wildcard [**] 216.150.152.145:137 ->
EDU.NET.5.44:137
11/08-04:33:10.393374 [**] SMB Name Wildcard [**] 216.150.152.145:137 ->
EDU.NET.5.44:137
11/08-04:34:05.519681 [**] SMB Name Wildcard [**] 216.150.152.145:137 ->
EDU.NET.5.44:137
11/08-04:34:07.767903 [**] ICMP Echo Request L3retriever Ping [**] 216.150.152.145
-> EDU.NET.5.44
```

A majority of these queries occurred from (Nov 7 11:39 to 11-8 12:35) and (Nov 8 21:01 to Nov 9 11:10), with 25 queries occurring between 12:34 and 14:03 on Nov 8.

The purpose for the interlaced L3retriever pings is unclear, but the long durations of scanning a single host may indicate an attacker trying to locate a specific user or capture a list of users logging into the machine.

The remaining machines with high queries are all from your university; and only scanned addresses within your network. These are not cause for alarm, but may be worth watching for continued activity.

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
EDU.NET.163.53	3033	3033	1800	1800
EDU.NET.239.78	2329	2329	1429	1429

EDU.NET.233.126	1829	1829	1138	1138
EDU.NET.217.42	1550	1550	951	951
EDU.NET.230.142	1221	1221	759	759
EDU.NET.205.114	1177	1177	761	761
EDU.NET.219.198	965	965	353	353
EDU.NET.219.102	947	947	404	404
EDU.NET.85.111	933	933	397	397
EDU.NET.203.206	830	830	522	522

5.10. SCAN Proxy Attempt

Signature	# Alerts	# Sources	# Destinations
SCAN Proxy attempt	20203	250	11581

The Scan Proxy events are signs that someone may be attempting to locate proxy servers within your network. If a proxy server is located within your network, an attacker could launch attacks or hide their activity, making it appear to originate from your IP address. False positives can indicate a server offering services from this port, which is a common alternate web services port.

The following signatures match the event label:

alert tcp $EXTERNAL_NET any -> $HOME_NET 1080 (msg:"SCAN Proxy attempt";flags:S; classtype:attempted-recon; sid:615; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 8080 (msg:"SCAN Proxy attempt";flags:S; classtype:attempted-recon; sid:620; rev:1;)

The top two destinations indicate traffic patters consistent with web usage. Many of the events are in short bursts within a few seconds, across “sessions” which stretch several minutes. These are consistent with multiple GET connections issues by a browser that is not using HTTP 1.1 persistent connections. These two hosts are likely hosting web services on port 8080, or infrequently being used as a proxy.

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
EDU.NET.253.105	607	1267	34	103
EDU.NET.53.45	227	4893	14	25

It is worth pointing out that these two destinations are also the recipients of many other recorded events, largely "MISC Large UDP Packets". These include:

- 4247 instances of MISC Large UDP Packet
- 378 instances of Incomplete Packet Fragments Discarded
- 371 instances of INFO FTP anonymous FTP
- 288 instances of FTP passwd attempt
- 28 instances of INFO MSN IM Chat data
- 3 instances of SMB Name Wildcard
- 2 instances of INFO - Possible Squid Scan
- 2 instances of EXPLOIT x86 NOOP
- 2 instances of FTP STOR 1MB possible warez site

The top 100 sources do appear to indicate scans, as nearly every recorded event under these sources indicates a new destination in the same /24 netmask. None of these are believed to pose a serious threat, since no single source scanned more than 500 destination IP's within your network.

5.11. Incomplete Packet Fragments Discarded

Signature	# Alerts	# Sources	# Destinations
Incomplete Packet Fragments Discarded	20105	14	26

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

Not surprisingly, the top three sources of this alert are also the top three sources of the "Misc Large UDP Packet" in discussed in section 5.3. These three sources account for 96.7% (19,440 of 20,105) of the events recorded. The fourth is included in the table to indicate the sharp drop-off in event counts per source.

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
61.150.5.19	12096	26014	7	9
61.134.9.88	6290	32880	8	22
61.150.5.18	1054	22059	2	7
218.2.4.101	32	38	1	1

Of the sources that were engaged in sending large UDP packet transmissions into your networks, nearly all are intertwined with simultaneous alerts for "Incomplete Packet Fragments Discarded". The events below clearly portray the relationship to the Large

UDP Traffic. The source/destination port of 0 is believed to be caused by the preprocessor logging, which does not report on source/destination ports. It is possible that these are not unique traffic types, and that your Snort sensors just did not receive all fragments.

```
11/10-15:11:43.133118 [**] MISC Large UDP Packet [**] 61.150.5.19:1327 ->
EDU.NET.153.189:1621
11/10-15:11:43.358313 [**] MISC Large UDP Packet [**] 61.150.5.19:1327 ->
EDU.NET.153.189:1621
11/10-15:11:44.248401 [**] Incomplete Packet Fragments Discarded [**]
61.150.5.19:0 -> EDU.NET.153.189:0
11/10-15:11:45.129209 [**] MISC Large UDP Packet [**] 61.150.5.19:1327 ->
EDU.NET.153.189:1621
11/10-15:11:45.243725 [**] MISC Large UDP Packet [**] 61.150.5.19:1327 ->
EDU.NET.153.189:1621
11/10-15:11:45.449999 [**] Incomplete Packet Fragments Discarded [**]
61.150.5.19:0 -> EDU.NET.153.189:0
11/10-15:11:46.843170 [**] Incomplete Packet Fragments Discarded [**]
61.150.5.19:0 -> EDU.NET.153.189:0
11/10-15:11:46.950354 [**] MISC Large UDP Packet [**] 61.150.5.19:1327 ->
EDU.NET.153.189:1621
11/10-15:11:47.135269 [**] MISC Large UDP Packet [**] 61.150.5.19:1327 ->
EDU.NET.153.189:1621
11/10-15:11:47.544668 [**] Incomplete Packet Fragments Discarded [**]
61.150.5.19:0 -> EDU.NET.153.189:0
11/10-15:11:48.342542 [**] Incomplete Packet Fragments Discarded [**]
61.150.5.19:0 -> EDU.NET.153.189:0
11/10-15:11:49.247996 [**] Incomplete Packet Fragments Discarded [**]
61.150.5.19:0 -> EDU.NET.153.189:0
11/10-15:11:50.635023 [**] MISC Large UDP Packet [**] 61.150.5.19:1327 ->
EDU.NET.153.189:1621
```

5.12. ICMP Echo Request – BSD Type

Signature	# Alerts	# Sources	# Destinations
ICMP Echo Request BSDtype	16018	26	24

These events indicate that a BSD system outside your network pinged a host within your network. The packet source is identified as a BSD host by matching the payload string shown in the rule below.

Te following rule was triggered:
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING BSDtype"; itype:8; content:"|08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17|"; depth:32; reference:arachnids,152; sid:368; classtype:misc-activity; rev:4;)

This signature, itself, does not identify an attack, but could identify attempts to map a network. In lieu of the absence of ICMP echo requests from non-BSD Operating Systems, the fact that these are identified as originating from BSD hosts does not raise any suspicion. Because BSD hosts comprise a small percentage of hosts on the internet (<2%), it is likely that your IDS implementation does not alert on echo requests from most operating systems.

Only one host (EDU.NET.70.148 – mirrors.YOUR.edu) received these pings from more than 2 external hosts and exceeded 25 pings in total. Judging from the hostname and miscellaneous ftp events recorded, this host is apparently an FTP mirror site. Additionally, the top 10 sources of this alert were also connected to this server via anonymous FTP. The pattern indcates that the anonymous users on the mirrors site may have been attempting to identify network bottlenecks during slow downloads. No other BSD ping events in this log correlated with other suspicious events or raised further concern.

5.13. Possible Red Worm Traffic

Signature	# Alerts	# Sources	# Destinations
High port 65535 udp - possible Red Worm – traffic	8165	45	52

The Red Worm is another name for the well known Adore Worm that was found in the wild in late Larch of 2001. This worm infects Linux systems in a similar fashion as the Li0n and Ramen worms by scanning for multiple vulnerabilities to exploit. Adore tests LPRng, rpc-statd, wu-ftpd, and BIND for vulnerable versions and exploits the vulnerable versions.

Adore adds and replaces several binaries during the infection process and installs a backdoor that allows shell access as root on the infected system. The availability of the backdoor is announced to the following email addresses (adore9000@21cn.com, adore9000@sina.com, adore9001@21cn.com, adore9001@sina.com) and can be located by scanning for the backdoor. The backdoor is provided by the trojan Kernel Log Daemon, which replaces the previously installed daemon.

Snort rules indicating Adore/Red worm backdoor access or traffic have not been released with a message matching the signature above, and reports dispute that the backdoor operates over UDP. If the reports are correct, it is possible that all events logged are false alerts.

Adore/Red Worm (trojan'ed klogd)
Also message replaces the "Kernel to logger" (**klogd**), by a program of the backdoor type that uses **ICMP** instead of **TCP** or **UDP** . This backdoor allows to the access to **root shell** , through port **65535**

> According to Anthony Dell –
> http://rr.sans.org/threats/mutation.php
> Once the *klogd* program (originally called *icmp*) is executed, it listens for an ICMP packet that is 77 bytes in length. Once it has received a packet of proper length, it binds a socket to TCP port 65535 which then allows root access to anyone telnetting to that port.

According to Clifford Yago -
http://www.sans.org/y2k/practical/Clifford_Yago_GCIA.doc

> The icmp "sekure ping backdoor" package is compiled. Sekure ping backdoor provides a root shell to allow connections when an echo request ICMP packet of a certain datagram size is directed at a specific port.
>
> Two macros in the icmp.c file reveal these requirements:
>
> ```
> #define SIZEPACK 77
> #define PORT     65535
> ```
>
> A ping directed at port 65535 with a packet size of 77 bytes will make the sekure ping backdoor operational.

If the Snort rule triggered on either source or destination ports 65535, some false positives would be expected. Several dozen sources and destinations were identified, but none of the traffic could equate to any type of meaningful session (assumed to be greater than 40 packets) except for one remote destination.

The "session" occurred between 00:05 and 00:37 on Nov-11 between EDU.NET.98.178 (no DNS resolution) and the suspected victim 66.79.17.223 (66-79-17-223.coastalnow.net). The first and last 3 events logged are shown below to indicate times, ports, and patterns.

```
11/11-00:05:35.401079 [**] High port 65535 udp - possible Red Worm - traffic [**] EDU.NET.98.178:6112 -> 66.79.17.223:65535
11/11-00:05:40.320008 [**] High port 65535 udp - possible Red Worm - traffic [**] EDU.NET.98.178:6112 -> 66.79.17.223:65535
11/11-00:05:40.520246 [**] High port 65535 udp - possible Red Worm - traffic
```

```
[**] EDU.NET.98.178:6112 -> 66.79.17.223:65535
----------------------------------------
11/11-00:37:27.151295 [**] High port 65535 udp - possible Red Worm -
traffic [**] EDU.NET.98.178:6112 -> 66.79.17.223:65535
11/11-00:37:28.073617 [**] High port 65535 udp - possible Red Worm -
traffic [**] EDU.NET.98.178:6112 -> 66.79.17.223:65535
11/11-00:37:28.848912 [**] High port 65535 udp - possible Red Worm -
traffic [**] EDU.NET.98.178:6112 -> 66.79.17.223:65535
```

Neither host was identified in any other events logged during the 5 days. While the information here is suspicious, we would recommend gathering more information before stating that this is likely an attack that originated from within your network.

5.14. Fragment Reassembly Time Exceeded

Signature	# Alerts	# Sources	# Destinations
ICMP Fragment Reassembly Time Exceeded	6310	69	74

This is an ICMP error that is returned by a host that did not receive all fragments advertised. Some events are expected, as traffic can be dropped or lost while traversing the Internet. If an excess of alerts point to a source or destination, it may indicate a fragmentation attack. Fragmentation attacks target TCP/IP stacks that mis-handle out of spec traffic.

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Fragment Reassembly Time Exceeded"; itype: 11; icode: 1; sid:410; classtype:misc-activity; rev:4;)

While there are several indications of long strings of reassembly time exceeded errors, none of the sources or destinations repeats the errors for more than 2 other source/destinations. If these increase, we would recommend capturing ICMP error packets with full alerting. This would allow you to capture information about the traffic that stimulated this response.

5.15. ICMP Destination Unreachable – Host Unreachable

Signature	# Alerts	# Sources	# Destinations
ICMP Destination Unreachable (Host Unreachable)	5484	370	62

This ICMP message (Code 1) is sent by a router when a destination host cannot be resolved by ARP on a subnet that is local to the router. Like error messages, the source of the ICMP message indicates the router that could not resolve the host, and the destination of the ICMP error identifies the host that sent a packet to the host that could not be resolved.

Receiving this error can indicate (the last two are malicious):

- Your hosts initiated a connection to a host that is not online (or does not exist)

- Your machines responded to traffic from a source that has dropped offline.
- Your machines are responding to traffic with a forged source address.
- That a forged ICMP error has been sent to interrupt your traffic flow.

The triggered rule is:
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Destination Unreachable (Host Unreachable)"; itype: 3; icode: 1; sid:399; classtype:misc-activity; rev:4;)

There are no patterns to this traffic that indicate malicious traffic, such as extended durations of error receipt or simultaneous receipts of errors from several sources.

5.16. Watchlist 000222 NET-NCFC

Signature	# Alerts	# Sources	# Destinations
Watchlist 000222 NET-NCFC	3271	18	12

These logs are generated by traffic with a source IP indicating origination from "The Computer Network Center Chinese Academy of Sciences" in Beijing China. The rule appears to log any source IP in the 159.226.0.0/16 network. The following are the 18 sources identified:

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
159.226.41.166	1083	1083	1	1
159.226.45.204	841	841	1	1
159.226.159.146	693	693	2	2
159.226.118.89	291	296	1	1
159.226.99.2	96	96	1	1
159.226.39.171	80	80	1	1
159.226.21.3	64	65	1	2
159.226.42.11	52	52	1	1
159.226.61.238	21	21	1	1
159.226.40.195	15	18	1	1
159.226.21.30	12	12	1	1
159.226.45.3	7	11	1	3
159.226.250.54	5	5	1	1
159.226.228.1	4	4	1	1

159.226.159.152	2	2	1	1
159.226.59.123	2	2	1	1
159.226.64.223	2	2	1	1
159.226.205.4	1	1	1	1

Five of these hosts also triggered other alerts- which are summarized below:

```
11/09-07:04:42.268847 [**] WEB-MISC prefix-get // [**] 159.226.118.89:4926 ->
EDU.NET.253.114:80
11/09-07:09:33.165332 [**] WEB-MISC prefix-get // [**]
159.226.118.89:4951 -> EDU.NET.253.114:80
11/09-07:09:35.869101 [**] WEB-MISC prefix-get // [**]
159.226.118.89:4954 -> EDU.NET.253.114:80
11/09-07:09:37.007731 [**] WEB-MISC prefix-get // [**]
159.226.118.89:4955 -> EDU.NET.253.114:80
11/09-07:09:37.814800 [**] WEB-MISC prefix-get // [**]
159.226.118.89:4956 -> EDU.NET.253.114:80
11/11-22:27:02.565946 [**] MISC source port 53 to <1024 [**]
159.226.21.3:53 -> EDU.NET.1.4:53
11/10-02:44:46.410804 [**] CS WEBSERVER - external web traffic [**]
159.226.40.195:3414 -> EDU.NET.100.165:80
11/10-02:44:59.126002 [**] CS WEBSERVER - external web traffic [**]
159.226.40.195:3423 -> EDU.NET.100.165:80
11/10-02:45:19.619646 [**] CS WEBSERVER - external web traffic [**]
159.226.40.195:3430 -> EDU.NET.100.165:80
11/07-22:00:24.463742 [**] MISC source port 53 to <1024 [**]
159.226.45.3:53 -> EDU.NET.1.3:53
11/08-22:18:28.068010 [**] MISC source port 53 to <1024 [**]
159.226.45.3:53 -> EDU.NET.1.5:53
11/09-14:20:30.080310 [**] MISC source port 53 to <1024 [**]
159.226.45.3:53 -> EDU.NET.1.3:53
11/11-22:45:55.003757 [**] MISC source port 53 to <1024 [**]
159.226.45.3:53 -> EDU.NET.1.5:53
```

About half of these are from web traffic; With requests and replies originating from the NCFC network and your network. The top two sources were apparently engaged in telnet sessions. The first session originated from your network to a telnet server in NCFC on 11-7 09:32, and closed at 17:52:10. The first and last events are below:

```
11/07-09:32:56.214670 [**] Watchlist 000222 NET-NCFC [**] 159.226.41.166:23 ->
EDU.NET.163.238:4916
```

```
11/07-17:52:10.305751 [**] Watchlist 000222 NET-NCFC [**] 159.226.41.166:23 ->
EDU.NET.163.238:4916
```

While 1083 packets received is rather low for an 8 ½ hour telnet session, there are no gaps longer than 2 minutes. This is believed to be a single session, since the source port does not change. The traffic was irregular, with bursts, indicating some activity.

The second session originated from NCFC to a telnet server in your network on 11-7 22:00, and closed at 22:46. The first and last events are below:

```
11/07-22:00:27.042669 [**] Watchlist 000222 NET-NCFC [**] 159.226.45.204:1478 ->
EDU.NET.6.7:23
```

```
11/08-22:46:16.006829 [**] Watchlist 000222 NET-NCFC [**] 159.226.45.204:1632 ->
EDU.NET.6.7:23
```

If there are no legitimate reasons for NCFC to be sending or receiving packets to/from your network, full alerting or telnet session logging could provide further information.

5.17. SYN_FIN Scan

Signature	# Alerts	# Sources	# Destinations
SYN-FIN scan!	3068	1	3068

Once considered a "stealth scan", these events identify one of the noisiest scanning attempts from a single host found within these logs. A SYN-FIN scan sends TCP packets with the SYN and FIN flags set; which should not occur in normal traffic. Many systems will reply with ACK-RST bits set of the port is closed and SYN-ACK if open.

This type of scanning is easily picked up, and scans this blatant indicate carelessness or inexperience on the behalf of the attacker. As with any obvious scan for specific services, the attacker is likely to have a toolkit ready to exploit vulnerable machines found.

The scanning host (200.254.62.75 – no DNS resolution) was scanning for SSH daemons. It is possible that this host attempted to exploit vulnerabilities found in earlier SSHd releases. If your Snort implementation did not have rules to identify SSHd exploits, your systems with SSH daemons prior to 2.2 should be checked for rootkits or signs of being exploited.

5.18. Echo Request - NMAP or HPING2

Signature	# Alerts	# Sources	# Destinations
ICMP Echo Request Nmap or HPING2	3033	53	383

These events are logged when a host outside your network sends an ICMP echo request to a destination within your network. What makes the request unique is that it does not

have a payload. Mot TCP/IP stacks include a payload, and the absence may indicate that the packet was "crafted". The NMAP or HPING2 label infers that the request was crafted by a tool. These tools are not definite sources of the packet, but are capable of duplicating this traffic.

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Nmap2.36BETA or HPING2 Echo ";itype:8;dsize:0; reference:arachnids,162; classtype:attempted-recon; sid:468; rev:1;)

The top destination, 149.1.1.1 is known to be associated with the Timesink ADbot, which is included in several shareware programs. This is an often discussed, yet little understood piece of "spyware". This ADbot appears to ping 149.1.1.1 with an empty payload approximately every half hour. The IP resolves to a netblock (149.1.1.1-141.255.255.255) owned by PSInet in ARIN, but does not resolve in DNS.

A discussion thread regarding the ICMP traffic can be found starting at: http://archives.neohapsis.com/archives/sf/ms/2000-q3/0120.html

5.19. GNUTella Connect Accept

Signature	# Alerts	# Sources	# Destinations
INFO Inbound GNUTella Connect accept	3032	42	2311

These events log that hosts within your network accepted a GNUTella connection from a host outside your network. GNUTella is a peer-to-peer file sharing application that can be used without requiring a central server. Just as with other "INFO" logs, the events logged by the following rule do not indicate an intrusion attempt, but identify a specific type of traffic.

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"INFO Outbound GNUTella Connect accept"; content: "GNUTELLA OK"; nocase; depth: 40; classtype:bad-unknown; sid:558; rev:1;)

None of the GNUTella Connect accepts appears suspicious or relates to suspicious events.

5.20. SMTP Relaying Denied

Signature	# Alerts	# Sources	# Destinations
SMTP relaying denied	1777	7	20

These events indicate that an email was sent to your SMTP servers, where the recipient for the message existed outside of the domain the server was intended to handle messaging for. If the machine was improperly configured, and operating as an "open relay", it would then relay the email to the recipient's SMTP server. Open relays are sought by spammers because it reduces their bandwidth requirements by forcing someone else's servers (and bandwidth) to deal with sending the same message to multiple recipients. Open relays can also be used to hide the true source of emails, providing anonymity to black-hat senders.

Since Snort triggers on the error message, it only logs when the relay failed. Any relay attempts that were successful will not be captured. As with any error messaging, the source of the log will indicate the error generator (SMTP server that refused to relay), and the destination will indicate the host attempting to relay through the server.

The rule triggered was:
alert tcp $SMTP 25 -> $EXTERNAL_NET any (msg:"SMTP relaying denied"; flags: A+; content: "550 5.7.1"; depth:70; reference:arachnids,249; classtype:bad-unknown; sid:567; rev:4;)

The following "sources" within your network refused to relay email.

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
EDU.NET.253.51	1699	1706	6	9
EDU.NET.253.53	48	50	7	8
EDU.NET.253.52	24	28	6	8
EDU.NET.253.43	2	3	2	3
EDU.NET.100.230	2	3	2	3
EDU.NET.162.64	1	1	1	1
EDU.NET.253.42	1	2	1	2

The following "destinations" are the 3 top offenders of 20 logged:

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
134.192.74.162	1689	1689	1	1
129.71.35.30	47	47	2	2
66.66.152.225	10	11	1	1

One "destination" (source of relay attempts) stands out due to the number of relaying attempts tried. The system attempting 95% (1689 of 1777) of the relay attempts does not resolve in DNS, but resolves to the University of Maryland at Baltimore in ARIN. We cannot derive what domains were tried, but the attempts were persistent. The attempts came during two blocks of time, each on different days. The fact that more than one attempt could be tried per second indicates that these attempts were scripted.
First Block Nov-7 15:10-16:20
Second block Nov-9 15:07-16:50

The remaining hosts that attempted relays appear to have been manually operated, as the timing between attempts was, t a minimum, several seconds- with no regularity in timing.

6. Top Talkers

The top talker lists were generated in a manner that would show the hosts generating the highest number of alerts, the number of alert types triggered by the host, and its contribution to the count of the top 20 event types reviewed above.
The Columns indicate:

- Rank: Ordered from one to ten, in decreasing numbers
- Source: The source of the alert. (* Response alert destinations were included to indicate that this host address was identified as the source of the stimulus)
- # Alerts: Total number of scans and alerts for this host
- # Alert Types: The number of unique alert types triggered by this host

The last two columns indicate the number any type of events that were included in the top 20 alert types reviewed above. Hosts that were indicated as being a source of more than one of the top 20 events by type will have additional right hand rows to indicate the additional alert types.

Top Ten Internal Talkers

Rank	Source	# Alerts	# Alert Types	# And Type of Alerts in top 20 (by type)	
1	EDU.NET.140.9	44280	7	1316	* ICMP Destination Unrch (host unreachable)
2	EDU.NET.70.148	16740	16	11	* ICMP Destination Unrch (host unreachable)
3	EDU.NET.98.178	4308	2	4306	High Port 65535 - Possible Red Worm
4	EDU.NET.163.53	3033	1	3033	SMB Name Wildcard
5	EDU.NET.70.64	2836	10	1859	* ICMP Destination Unrch (host unreachable)
6	EDU.NET.239.78	2329	1	2329	SMB Name Wildcard
7	EDU.NET.99.39	2110	6	2097	Inbound GNUTella Connect Accept (allow share)
8	EDU.NET.233.126	1829	1	1829	SMB Name Wildcard
9	EDU.NET.217.42	1550	1	1550	SMB Name Wildcard
10	EDU.NET.70.11	1298	12	733	* ICMP Destination Unrch (host unreachable)

* Indicates that this source IP provided the stimulus for the ICMP error

Top Ten External Talkers

Rank	Source	# Alerts	# Alert Types	# And Type of Alerts in top 20 (by type)	
1	61.134.9.88	32880	3	26578	Large UDP Packet
				6290	Incomplete packet fragments discarded
				12	High Port 65535 UDP - Possible Red Worm
2	61.150.5.19	26014	3	13917	Large UDP Packet
				12096	Incomplete packet fragments discarded
3	61.150.5.18	22059	2	21005	Large UDP Packet
				1054	Incomplete packet fragments discarded
4	61.175.133.20	7945	2	7944	Large UDP Packet
5	216.150.152.145	7702	2	3895	SMB Name Wildcard
6	61.153.17.24	6115	1	6115	Large UDP Packet
7	61.153.116.195	3949	1	3949	Large UDP Packet
8	66.79.17.223	3728	1	3728	High Port 65535 - Possible Red Worm
9	200.254.62.75	3068	1	3068	SYN-FIN scan
10	128.223.4.21	2618	3	2515	Ping - BSD Type

7. Top Talker Registration Information

These seven hosts were chosen because they generated or received traffic that raised the most attention. Our recommendation would be to create a watchlist rule for the netblocks containing the following suspect addresses. The following registration will assist in building watchlists and gathering contact information.

This host generated the largest number of "Misc Large UDP" and is suspected of scanning for Red Worm backdoors

whois -h whois.apnic.net 61.134.9.88
[whois.apnic.net]

% Rights restricted by copyright. See http://www.apnic.net/db/dbcopyright.html
% (whois6.apnic.net)

inetnum: 61.134.3.0 - 61.134.20.95
netname: SNXIAN
descr: XI'AN DATA BUREAU
country: CN
admin-c: WWN1-AP
tech-c: WWN1-AP
mnt-by: MAINT-CHINANET-SHAANXI
mnt-lower: MAINT-CN-SNXIAN
changed: ipadm@public.xa.sn.cn 20010427
source: APNIC

person: WANG WEI NA
address: Xi Xin street 90# XIAN
country: CN
phone: +8629-724-1554
fax-no: +8629-324-4305
e-mail: xaipadm@public.xa.sn.cn
nic-hdl: WWN1-AP
mnt-by: MAINT-CN-SNXIAN
changed: wwn@public.xa.sn.cn 20001127
source: APNIC

These hosts generated the second and third largest number of "Misc Large UDP" alerts

whois -h whois.apnic.net 61.150.5.19 and 61.150.5.18
[whois.apnic.net]

% Rights restricted by copyright. See http://www.apnic.net/db/dbcopyright.html
% (whois7.apnic.net)

inetnum: 61.150.0.0 - 61.150.31.255
netname: SNXIAN

descr: xi'an data branch,XIAN CITY SHAANXI PROVINCE
country: CN
admin-c: WWN1-AP
tech-c: WWN1-AP
mnt-by: MAINT-CHINANET-SHAANXI
mnt-lower: MAINT-CN-SNXIAN
changed: ipadm@public.xa.sn.cn 20010309
source: APNIC

person: WANG WEI NA
address: Xi Xin street 90# XIAN
country: CN
phone: +8629-724-1554
fax-no: +8629-324-4305
e-mail: xaipadm@public.xa.sn.cn
nic-hdl: WWN1-AP
mnt-by: MAINT-CN-SNXIAN
changed: wwn@public.xa.sn.cn 20001127
source: APNIC

This host generated the fourth largest number of "Misc Large UDP" alerts
whois -h whois.apnic.net 61.175.133.20
[whois.apnic.net]

% Rights restricted by copyright. See http://www.apnic.net/db/dbcopyright.html
% (whois6.apnic.net)

inetnum: 61.174.0.0 - 61.175.255.255
netname: CHINANET-ZJ
descr: CHINANET Zhejiang province network
descr: Data Communication Division
descr: China Telecom
country: CN
admin-c: CH93-AP
tech-c: CZ61-AP
mnt-by: MAINT-CHINANET
mnt-lower: MAINT-CHINANET-ZJ
changed: hostmaster@ns.chinanet.cn.net 20010406
source: APNIC

person: Chinanet Hostmaster
address: A12,Xin-Jie-Kou-Wai Street
country: CN
phone: +86-10-62370437
fax-no: +86-10-62053995
e-mail: hostmaster@ns.chinanet.cn.net

nic-hdl: CH93-AP
mnt-by: MAINT-CHINANET
changed: hostmaster@ns.chinanet.cn.net 20000101
source: APNIC

person: CHINANET ZJMASTER
address: no 378,yan an road,hangzhou,zhejiang
country: CN
phone: +86-571-7015441
fax-no: +86-571-7027816
e-mail: master@dcb.hz.zj.cn
nic-hdl: CZ61-AP
mnt-by: MAINT-CHINANET-ZJ
changed: master@dcb.hz.zj.cn 20001219
source: APNIC

The following is the greatest source of SMB Name Wildcards.

whois -h whois.arin.net 216.150.152.145
(whois -h whois.arin.net NET-XAND-BLK-1)
[whois.arin.net]
Xand Corporation (NET-XAND-BLK-1)
11 Skyline Drive
Hawthorne, NY 10532
US

Netname: XAND-BLK-1
Netblock: 216.150.128.0 - 216.150.159.255
Maintainer: XAND

Coordinator:
Xand Corporation (ZX8-ARIN) dnsadmin@xand.com
914-592-8282

Domain System inverse mapping provided by:

AUTH01.DNS.XAND.COM 216.150.131.196
AUTH02.DNS.XAND.COM 216.150.131.197

The following host received traffic suspected of targeting a Red Worm backdoor.

whois -h whois.arin.net 66.79.17.223
[whois.arin.net]
Mebtel Communications (NETBLK-MEBTEL-BLK-3)
103 South Fifth Street
Mebane, NC 27302
US

Netname: MEBTEL-BLK-3
Netblock: 66.79.0.0 - 66.79.95.255
Maintainer: MEBT

Coordinator:
REITER, DENNIS (DR666-ARIN) REITERD@GALLATINRIVER.COM
3093455261

Domain System inverse mapping provided by:

DNS0.MEBTEL.NET 208.241.20.25
DNS1.MEBTEL.NET 208.241.20.26

ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

This host performed a SYN-FIN scan for SSH daemons on 3068 of your hosts.
whois -h whois.nic.br 200.254.62.75
[whois.nic.br]

% Copyright registro.br
% The data below is provided for information purposes
% and to assist persons in obtaining information about or
% related to domain name and IP number registrations
% By submitting a whois query, you agree to use this data
% only for lawful purposes.
% 2002-02-04 00:13:32 (BRST -02:00)

inetnum: 200.254/16
aut-num: AS4230
abuse-c: GSE6
owner: EMBRATEL-EMPRESA BRASILEIRA DE TELECOMUNICAÇÕES SA
ownerid: 033.530.486/0001-29
responsible: Ricardo S. Maceira
address: Av. Presidente Vargas, 1012,
address: 20179-900 - Rio de Janeiro - RJ
phone: (021) 2519-8729 []
owner-c: RSM3
tech-c: CAP12
inetrev: 200.254.62/24
nserver: NS.EMBRATEL.NET.BR
nsstat: 19991213 AA
nslastaa: 19991213

nic-hdl-br: CAP12
person: Gerencia Técnica de Operações Internet
e-mail: hostmaster@EMBRATEL.NET.BR

address: Rua Senador Pompeu, 119, 6 and
address: 20080-001 - Rio de Janeiro - RJ
phone: (021) 5192507 []
created: 19980202
changed: 20020108

nic-hdl-br: GSE6
person: Grupo de Segurança Internet da Embratel
e-mail: abuse@EMBRATEL.NET.BR
address: R. Senador Pompeu, 119, 6. andar
address: 20080-001 - Rio de Janeiro - RJ
phone: (078) 21278 []
created: 20001005
changed: 20001005

nic-hdl-br: RSM3
person: Gerência do Backbone Internet EMBRATEL
e-mail: domain-admin@EMBRATEL.NET.BR
address: Rua Alexandre Mackenzie, 75, 6. andar
address: 20221-410 - Rio de Janeiro - RJ
phone: (021) 2519-7043 []
created: 19980123
changed: 20020107

remarks: Security issues should also be addressed to
remarks: nbso@nic.br, http://www.nic.br/nbso.html
remarks: Mail abuse issues should also be addressed to
remarks: mail-abuse@nic.br

% whois.registro.br accepts only direct match queries.
% Types of queries are: domains (.BR), BR POCs, CIDR blocks,
% IP and AS numbers.

---

8. OOS Analysis

Out-Of-Spec (OOS) packets are identified by combinations of flags that are improper ofr use on the Internet. Examples include SYN-FIN bits set on one packet, or Don't Fragment and More Fragments on another. There are several combinations of bits that can lead to a packet being OOS.

Of the 3723 recorded packets that were Out-of-Spec (OOS), most can be quickly divided into the following groups:

- 3098 SYN-FIN scans
- 482 21S flagged packets
- 46 DF MF bits set

- 107 Various other combinations of TCP flags

The 3098 SYN-FIN scans identify the same host that was identified in section 5.17 (200.254.62.75). Section 5.17 identifies 3068 SYN-FIN scans from source port 22 to destination port 22, but it is safe to assume that the IDS sensors lost a few packets while logging the events. Judging from the Alert and OOS logs, this is clearly a scan, and that these OOS packets were already "accounted for" in the alerts.

A majority of the remaining packets had one or both reserved bits set. However, Explicit Congestion Notification (ECN), which uses the previously "reserved" bits to provide network congestion notices, has recently been implemented in TCP/IP stacks of hosts and routers. Unfortunately, Snort's anomaly detection engine does not account for ECN implementations and will identify traffic with any combination of the previously reserved bits set as OOS.

One likely explanation for the large quantity of ECN bits 1 and 2 being flagged on SYN packets can be explained in the use of the Linux 2.4 Kernel. The 2.4 kernel now includes an option to enable ECN notification. If enabled, initial SYNs can be flagged with both bits set to request an aggressive response. Why aren't these seen more often? This capability is not pre-compiled into any Linux release at this date. An end user would need to recompile their kernel, requesting ECN, to enable this. The possibility exists that some of the 482 21S flagged packets are a result of kernel recompiles. Most of the 21S packets sent to/from the same hosts do not appear in rapid succession, such as in flooding or excessive retries. None of the "reserved bit set" packets appear consistent with a rapid attack of any type.

There are tools that can duplicate packets such as this. These include
TCP Traceroute http://packetstorm.widexs.nl/UNIX/security/tcptraceroute-1.2.tar.gz
HPING2 http://www.hping.org/
NMAP http://nmap.org/

The Don't Fragment and More Fragments flags should not exist together for obvious reasons. Either can be set, indicating either a request to not fragment- or stating that more fragments follow. Most hosts logged sending DF MF packets, sent one or two, and were not seen again during the logging period. Two hosts (64.108.76.25, and 64.165.71.53) did appear to send bad DF MF packets on more than one "session". Like all other DF MF packets, the source/destination ports were 0 and the Fragment size was indicated as 0x22. It is possible that these are from failing or corrupted stacks.

The remaining packets include a wide array of various bits. None appear in rapid succession, or in long strings from the same host. Without correlating these with other events, these can likely be viewed as victims of bad stacks or routers.

9. Compromised Internal Machines

Only one machine gives indication hinting at a compromise. This host (EDU.NET.98.178 – does not resolve in DNS) received traffic identified as possible Red Worm activity.

The suspicious "MISC Large UDP Packets" originating from addresses that resolve to China have raised enough attention that we would recommend checking the destination hosts listed in Section 5.3. The recommendation is to check for Trojans, viruses, or applications listening on any protocol on high ports.

10. Defensive Recommendations

Because of the excessive number of events being logged, your analysts are being overwhelmed by the volume of records. To maintain a defensive capability you must be able to monitor your logs within a reasonable amount of time to respond accordingly. The following are first step approaches to reducing your excessive false positives.

- Place your DHCP server(s) EDU.NET.5.75 and 76 into the portscan ignorehosts line. This will reduce nearly ¾'s of your scan logs.
- Place any remaining servers that initiate outbound connections, such as email servers, in your portscan ignorehosts line.
- Consider disabling informational logging if your policies do not restrict their policies. This will reduce unnecessary logging of chat data, GnuTella sharing, and online game sessions.
- Pass TCP port 53->53 traffic for DNS servers to reduce logging large name resolutions.
- Consider removing asymmetric logging that captures information that may be misleading. For example, if you are not logging default ICMP echo requests from most operating systems, do not log echo requests from BSD operating systems.
- If your MISC Traceroute rule is based solely on TTL, consider adding requirements such as port ranges or ICMP types to reduce false positives.
- Unless the CS webserver should not be contacted from outside hosts, remove the "external CS web traffic" rule and replace with more specific rules.

Fast alerting identifies patterns in traffic, but is not helpful for identifying details of suspect traffic. Logging detailed alerts (using -A full) for the following will help identify malicious traffic.

- ICMP errors. Full logging will include part of the packet providing the stimulus.
- MISC Large UDP. Since the traffic appears suspect, payload information will help discern the nature of the activity.
- Watchlist specific traffic. This will help clarify the activities of the suspected hosts.
- Any traffic from EDU.NET.98.178 with a destination port of 65535 (possible Red Worm)
- Any traffic to/from the hosts/netblocks addresses in section 7.

Increasing logging detail on specific alert groups can be done through running multiple instances of Snort per sensor. The recommended method is to load the rules so that any traffic type is reviewed by only one instance. This can be done by loading chains with traffic types. An example would be one instance of Snort performing full alerting on ICMP packets, and another fast logging TCP and UDP traffic.

We understand that, as a University, that you are restricted from placing firewalls at your perimeters. If within policy, consider filtering services that are typically not used to share information across the Internet such as NetBIOS services. If possible determine if hosts that are probing or attacking can be shunned. If this can be done, you may have an option to shun nonstandard services. Shunning standard services offers too much control to the attacker, allowing them to interfere with your normal traffic.

In lieu of the MISC Large UDP packets that appear suspicious, we recommend taking the some or all of the following hosts down to check for signs of Trojans, viruses, or rootkits.

- EDU.NET.111.221
- EDU.NET.84.195
- EDU.NET.53.40
- EDU.NET.53.45
- EDU.NET.53.49

Check your SMTP servers for presence of open relays. Seven systems refused to relay traffic. If you have servers SMTP servers not indicated in section 5.20, they should be checked.

11. Analysis Process

The largest portion of our data pre-processing was doe by Snortsnarf, which is available from Silicon Defense at http://www.silicondefense.com/software/snortsnarf.

Analysis of the data required some manual filtering before processing through an analysis engine. The Snortsnarf analysis tool used to group event types and identify hosts triggering multiple event types cannot handle the logs output by your version of Snort. Your release logs Internal IP addresses as MY.NET.XXX.XXX, where "MY.NET" obscures the first two octets of the IP address. Snortsnarf cannot parse alphabetical addresses, so

The scan logs contained an excessive number of false positives, which were removed and consolidated using:
cat scans.* |grep -v MY.NET.5.75:67 |grep -v MY.NET.5.76:67 > all.scans.log
Since the logs were separated by day and would not facilitate correlating across days, the scan and alert logs were combined into a single log before parsing through snortsnarf using: cat alert.* > snortsnarf.in.log and cat all.scans.log >> snortsnarf.in.log.
This allowed snortsnarf to parse all five days logs withoug requiring manual correlation between days. The payoff of combining logs is the requirement for additional RAM to

correlate the logs. Snortsnarf required a minimum of 1.5G of RAM to parse the remaining logs.

The Out Of Spec (OOS)logs are generated with several lines per entry, which ruled out grep –v filtering to eliminate events already logged (such as SYN-FIN scans). The approach taken was to grep and line count by categories to identify groups of packets. An example would be: cat oos.* |gep '21S' –c which yields 482 packets with both ECN (reserved) bits, SYN, and any other bits set.