Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst (GCIA)

Practical Assignment
Version 4.1

Michael Gauthier
SANS Network Security 2004 (September)
Submission date: March 21th, 2005

Abstract

This paper is a Security Audit Intrusion Report. It is based upon the following files that were retrieved from http://isc.sans.org/logs/. The /scans/scans.030916.gz file contained no data.

/oos/OOS_Report_2003_09_15_22271.gz
/oos/OOS_Report_2003_09_16_32593.gz
/oos/OOS_Report_2003_09_17_9362.gz
/oos/OOS_Report_2003_09_18_12914.gz
/oos/OOS_Report_2003_09_19_10419.gz
/scans/scans.030915.gz
/scans/scans.030916.gz
/scans/scans.030917.gz
/scans/scans.030918.gz
/scans/scans.030919.gz
/alerts/alert.030915.gz
/alerts/alert.030916.gz
/alerts/alert.030917.gz
/alerts/alert.030918.gz
/alerts/alert.030919.gz

The paper contains three main sections.

- Executive Summary
- Detailed Analysis
- Analysis Process

The primary purpose of the Executive Summary and Detailed Analysis sections is the fulfillment of the practical assignment objectives. The assignment states, “select the three most critical detects” this objective was weighted against an attempt not to duplicate work already done by others. Specifically the IRC/Botnet alerts appear to have received a great deal of attention already. The secondary factor was choosing detects which provided sufficient information to generate a meaningful analysis.

The Analysis Process section contains a detailed walkthrough of how the files were analyzed including commands run. It is intended to provide a starting point for future students so that their work may build upon these processes.

Table of Content

Executive Summary

This report analyzes three days worth of logs generated by the University's network intrusion detection system (NIDS). These logs contain 747,578 intrusion alerts covering 49 unique signatures. Previous reports have covered the dangers of botnets and their use of Internet relay chat (IRC) communication channels; this report will not duplicate that work.

After reviewing the NIDS logs one overarching problem becomes apparent, the system utilizes signatures that are to narrow in their focus. Numerous alerts analyzed here triggered based upon one criteria. In today's heterogeneous Internetworking environment there are very few malicious attacks that can be located based upon testing a single element of a packet.

In December of 2002 a Snort user list posting updated a network time protocol (NTP) rule to eliminate false positives caused by normal traffic. However even though these logs are from September of 2003 the rule still appears to be unchanged from the original.

The University needs to verify the integrity of the hosts identified in this report as having potentially been compromised. These hosts were identified as having responded to remote probes for backdoor programs. These programs would give remote users unfettered access to University resources as well as the ability to damage the University's reputation. Once this is completed it is imperative that the security team update the NIDS rulebase to current standards verifying that any rules identified here are corrected to test for multiple criteria before triggering an alert.

With an updated rulebase the University NIDS will provide its operators with a greatly increased probability of identifying attacks and intrusions. This will lead to a greater respect for its data and a higher level of trust from other information technology departments. Increased trust will inevitably lead to a more positive response and cooperation between departments leading to a higher level of security on the network.

Detailed Analysis

Files Analyzed

The following files were analyzed for this report. They were generated by various university systems between September 14th, 2003 and September 19th, 2003. Each file's contents ends on the evening of the data specified in the filename with the exception of the OOS logs. The OOS logs contain the previous day's log data.

```
/oos/OOS_Report_2003_09_15_22271.gz
/oos/OOS_Report_2003_09_16_32593.gz
/oos/OOS_Report_2003_09_17_9362.gz
/oos/OOS_Report_2003_09_18_12914.gz
/oos/OOS_Report_2003_09_19_10419.gz
/scans/scans.030915.gz
/scans/scans.030916.gz
/scans/scans.030917.gz
/scans/scans.030918.gz
/scans/scans.030919.gz
/alerts/alert.030915.gz
/alerts/alert.030916.gz
/alerts/alert.030917.gz
/alerts/alert.030918.gz
/alerts/alert.030919.gz
```

Relationship Identification

In a complex Internetwork environment there exists a number of systems many of which are interdependent. For the purpose of intrusion analysis it is important to identify stimulus and response. When reading a log entry we must identify what caused the entry to be created.

The scan logs show numerous entries for what appear to be outbound DNS queries from a local DNS server on IP address 130.85.1.3. We can verify the legitimacy of this traffic by identifying the destination with simple DNS queries. It is important to note that the owner of IP addresses can assign any name to a reverse DNS PTR record so we must also cross check these results with whois lookups to the various Internet registries to verify the ownership of the IP space.

```
echo 'select dstip,dstport,prot from scan where srcip LIKE "130.85.1.3" order by
dstip,dstport,prot;' | mysql gcia | uniq -c | sort -nr | head
  55787 192.26.92.30    53      UDP
  46178 203.20.52.5     53      UDP
  44379 130.94.6.10     53      UDP
  39550 192.52.178.30   53      UDP
  35848 205.231.29.244  53      UDP
  32312 131.118.254.34  53      UDP
  32067 192.55.83.30    53      UDP
  30574 192.148.252.171 53      UDP
  28321 205.231.29.240  53      UDP
  27894 63.251.136.209  53      UDP
host 192.26.92.30
30.92.26.192.in-addr.arpa domain name pointer c.gtld-servers.net.

whois -h whois.arin.net 192.26.92.30
```

```
OrgName:    VeriSign Global Registry Services
OrgID:      VGRS
Address:    21345 Ridgetop Circle
City:       Dulles
StateProv:  VA
PostalCode: 20166
Country:    US

NetRange:   192.26.92.0 - 192.26.92.255
CIDR:       192.26.92.0/24
NetName:    VGRSGTLD-3
NetHandle:  NET-192-26-92-0-1
Parent:     NET-192-0-0-0-0
NetType:    Direct Assignment
NameServer: L2.NSTLD.COM
NameServer: D2.NSTLD.COM
NameServer: E2.NSTLD.COM
NameServer: C2.NSTLD.COM
Comment:
RegDate:    2000-11-30
Updated:    2001-03-20

TechHandle: ZV22-ARIN
TechName:   VeriSign Global Registry Services
TechPhone:  +1-703-318-6444
TechEmail:  nstld@verisign-grs.com

OrgTechHandle: NETWO480-ARIN
OrgTechName:   Network Admin
OrgTechPhone:  +1-703-948-4300
OrgTechEmail:  netadmin@verisign.com

# ARIN WHOIS database, last updated 2005-03-05 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

While researching this traffic an unusual number of lookups was detected going to host “130.94.6.10” with the DNS PTR record “bsp1.bondedsender.org.” A whois lookup on the domain bondedsender.org did not return any noteworthy information but a quick look at there website returned the following information.

```
Overview of the Bonded Sender program

The growth of unsolicited email - spam - has negatively impacted both senders and
receivers of email. Traditional anti-spam systems have been developed to identify spam,
but these systems have the drawback of periodically destroying legitimate email traffic.

Sponsored by IronPort Systems, the Bonded Sender program turns the spam problem upside
down by identifying legitimate email traffic. Originators of legitimate email can now post
a financial bond to ensure the integrity of their email campaign. Receivers who feel they
have received an unsolicited email from a Bonded Sender can complain to their ISP,
enterprise, or IronPort and a financial charge is debited from the bond.

This market-based mechanism allows email senders to ensure their message gets to their end
user, and provides corporate IT managers and ISPs with an objective way to ensure only
unwanted messages get blocked. For FAQs and white papers describing the Bonded Sender
program, visit http://www.bondedsender.com

Configuring your spam filters to use Bonded Sender

Integrating Bonded Sender with email software is easy. Receivers of email can use the DNS
query commonly used by blacklist services to take advantage of the Bonded Sender program.
```

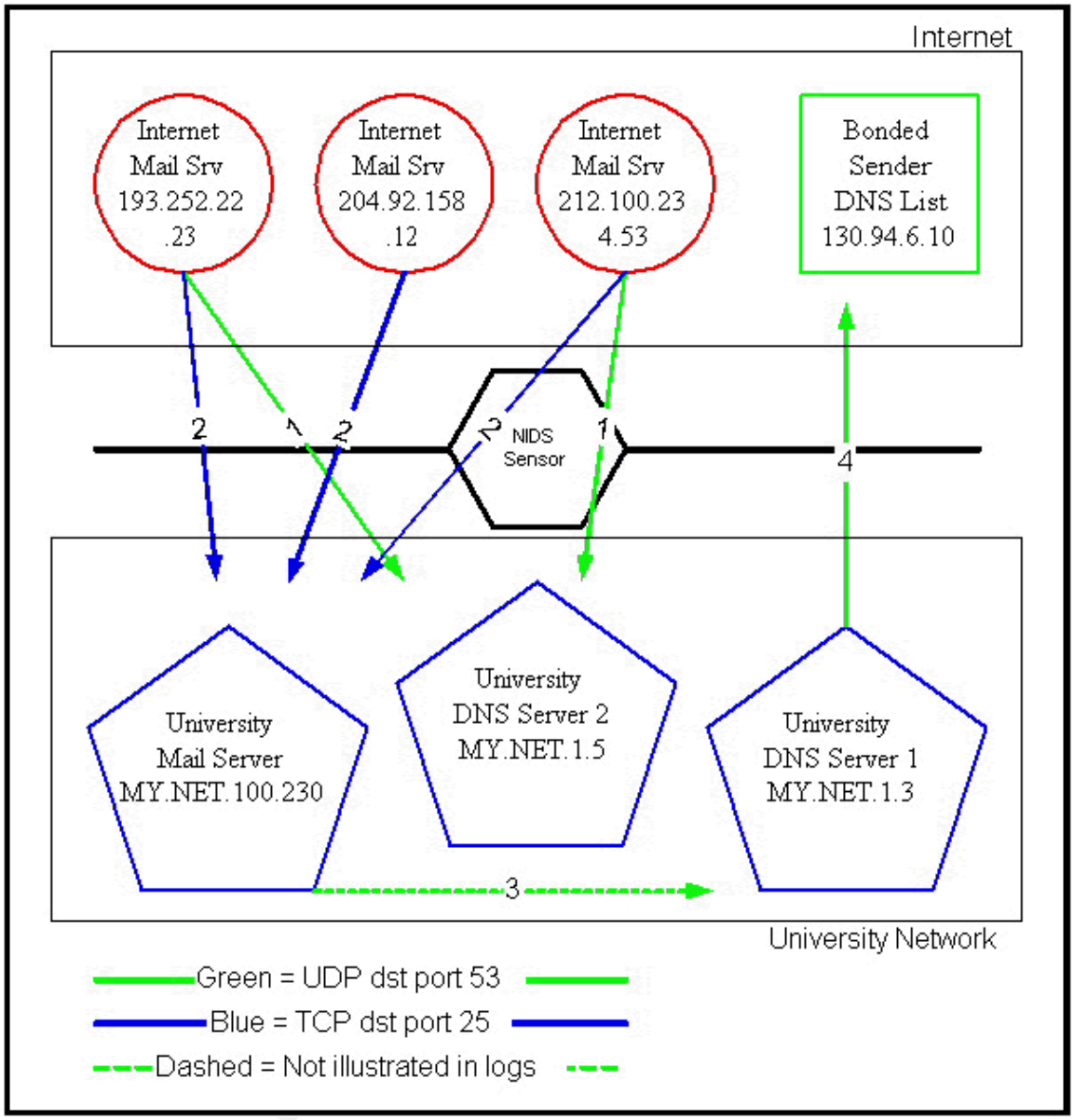
Further checks show that to query the bonded sender database the mail

software should be setup to send requests to sa.bondedsender.org an address that resolves to the IP address in question.

```
sa.bondedsender.org    A       IN      2700    130.94.6.10
```

It is extremely probable that a University mail server is causing the University DNS server to originate the traffic to bonded sender. Next we dig another layer deep to discover that the University mail server first receives SMTP email from the Internet causing it to query the University DNS server. Finally prior to the University mail server receiving mail the University's second DNS server receives a DNS request most likely an MX record lookup to locate the mail server's IP address. Following are the scan log entries and a link graph illustrating the traffic flow.

```
timeof               srcip        srcport dstip     dstport prot flags
2003-Sep-19 10:19:15 193.252.22.23  26370 130.85.1.5     53 UDP
2003-Sep-19 10:19:15 193.252.22.23  64141 130.85.100.230 25 TCP SYN 12****S* RESERVEDBITS
2003-Sep-19 10:19:16 130.85.1.3     38473 130.94.6.10    53 UDP

2003-Sep-19 20:30:17 193.252.22.23  45098 130.85.1.5     53 UDP
2003-Sep-19 20:30:17 193.252.22.23  62693 130.85.100.230 25 TCP SYN 12****S* RESERVEDBITS
2003-Sep-19 20:30:17 130.85.1.3     38473 130.94.6.10    53 UDP

2003-Sep-19 06:57:15 193.6.40.1     53    130.85.1.3     53 UDP
2003-Sep-19 06:57:15 193.6.40.1     3795  130.85.100.230 25 TCP SYN 12****S* RESERVEDBITS
2003-Sep-19 06:57:35 130.85.1.3     38473 130.94.6.10    53 UDP

NO INITIAL DNS
2003-Sep-15 08:13:14 204.92.158.12  36994 130.85.100.230 25 TCP SYN 12****S* RESERVEDBITS
2003-Sep-15 08:13:15 130.85.1.3     38473 130.94.6.10    53 UDP

NO INITIAL DNS
2003-Sep-18 16:59:09 204.92.158.12  51129 130.85.100.230 25 TCP SYN 12****S* RESERVEDBITS
2003-Sep-18 16:59:15 130.85.1.3     38473 130.94.6.10    53 UDP

2003-Sep-17 01:28:15 212.100.234.53 45290 130.85.1.5     53 UDP
2003-Sep-17 01:28:15 212.100.234.53 45291 130.85.100.230 25 TCP SYN 12****S* RESERVEDBITS
2003-Sep-17 01:28:20 130.85.1.3     38473 130.94.6.10    53 UDP

2003-Sep-18 01:02:12 212.100.234.53 62471 130.85.1.3     53 UDP
2003-Sep-18 01:02:12 212.100.234.53 62472 130.85.100.230 25 TCP SYN 12****S* RESERVEDBITS
2003-Sep-18 01:02:13 130.85.1.3     38473 130.94.6.10    53 UDP
```



The link graph displays a many to one relationship between the outside mail servers and the University DNS server. It also shows a many to one relationship between the outside mail servers and the University mail server. It is important to note that the presumed location of the network intrusion detection system (NIDS) sensor is at the perimeter of the network. We can assume this location because none of the logs show Intranetwork traffic. A one to one relationship between the University mail server and the University DNS server is inferred by our understanding of the client server DNS protocol. Finally a one to one relationship exists between the University DNS server and the bonded sender list server.

Detect Overview

A total of forty nine unique detects were logged in the alert logs over the time period reviewed.

```
Entries Alert
 644781 SMB Name Wildcard
  57074 257.257.30.4 activity
  14303 Incomplete Packet Fragments Discarded
  10269 High port 65535 udp - possible Red Worm - traffic
   9666 257.257.30.3 activity
   2324 connect to 515 from outside
   1810 ICMP SRC and DST outside network
   1755 SUNRPC highport access!
    986 NMAP TCP ping!
    982 Possible trojan server activity
    857 Null scan!
    791 EXPLOIT x86 NOOP
    517 TCP SRC and DST outside network
    443 High port 65535 tcp - possible Red Worm - traffic
    416 [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.
    382 [UMBC NIDS] External MiMail alert
    243 [UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC
    177 SNMP public access
    165 External RPC call
     96 EXPLOIT x86 stealth noop
     81 EXPLOIT NTPDX buffer overflow
     75 FTP passwd attempt
     70 FTP DoS ftpd globbing
     61 SMB C access
     52 EXPLOIT x86 setgid 0
     51 EXPLOIT x86 setuid 0
     30 RFB - Possible WinVNC - 010708-1
     21 TFTP - Internal UDP connection to external tftp server
     14 TFTP - External TCP connection to internal tftp server
     14 TCP SMTP Source Port traffic
     10 Tiny Fragments - Possible Hostile Activity
      8 Traffic from port 53 to port 123
      8 TFTP - External UDP connection to internal tftp server
      7 TFTP - Internal TCP connection to external tftp server
      6 Attempted Sun RPC high port access
      5 [UMBC NIDS IRC Alert] K:line d user detected, possible trojan.
      5 External FTP to HelpDesk 257.257.53.29
      3 Probable NMAP fingerprint attempt
      3 NIMDA - Attempt to execute cmd from campus host
      3 DDOS mstream client to handler
      2 [UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.
      2 NETBIOS NT NULL session
      1 SYN-FIN scan!
      1 Fragmentation Overflow Attack
      1 External FTP to HelpDesk 257.257.70.50
      1 External FTP to HelpDesk 257.257.70.49
      1 DDOS shaft client to handler
      1 Bugbear@MM virus in SMTP
      1 Back Orifice
```

Detect 1

```
timeof         action                          srcip       srcport dstip      dstport
09-16 06:03:47 EXPLOIT NTPDX buffer overflow 198.64.140.205 3186 257.257.97.104 123
```

This detect was chosen primarily because it is an exploit detect immediately followed by a remote control detect. No other alerts were noted coming from or going to either of the addresses associated with the NTPDX alert.

```
09-16 06:04:36 Back Orifice                198.64.140.205 12   257.257.97.104 31337
```

No entries were found for either address in the OOS logs. A number of scans were found targeting 257.257.97.104 and on September 17th and after there are a number of scans originating from 257.257.97.104. No entries were found in the scan log for 198.64.140.205.

September 15th through the 19th, 2003 were Monday through Friday therefore a weekend period does not explain the sudden start in scan activity from 257.257.97.104. The scans targeting 257.257.97.104 do not appear to have any direct connection to the NTPDX or Back Orifice alert. The scans target TCP ports 21 (FTP), 80 (WEB), 554 (RTSCP), and 34816 (MS03-018). Port 34816 is an exploit port for Microsoft Internet Information Server see reference MS03-018. The apparent culprit of most of the outgoing scans a peer to peer file sharing program called "Blubster" see SANS ISC notes at http://isc.sans.org/port_details.php?port=41170.

After researching the NTPDX exploit and the capabilities of Back Orifice they seem an unlikely combination. The NTP daemon would be found in most cases on a *IX system whereas Back Orifice is a remote control system for Windows. The scans for port 34816 for Microsoft IIS vulnerability (new as of May of 2003) also point to a possible Windows platform. The scans for port 41170 also point to a Windows platform use of the p2p app "Blubster" see there downloads section http://www.blubster.com/download/. Further examination of the end system will be required to make a definite determination.

Detect Generation

The University IDS system rules were not provided for correlation with this analysis nor were the actual packets logged that generated alerts. It is therefore not possible to be absolutely certain what traffic generated the alerts, however an educated guess can be made based upon what "standard" rules were known to exist at the time the logs were generated.

```
drop udp $EXTERNAL_NET any -> $HOME_NET 123 (msg:"EXPLOIT NTPDX buffer overflow"; dsize: >128; reference: arachnids,492;)
```

This rule can be found at http://xfiles.erin.utoronto.ca/pub/unix/security/hogwash/rules0727 in a file dated April of 2002. The rule is written in Snort syntax and the alert was likely generated by a Snort sensor. Snort versions 2.0.1 was released in July of 2003 according to the ChangLog file found at http://cvs.snort.org/viewcvs.cgi/snort/ChangeLog?rev=1.337.2.17&content-type=text/vnd.viewcvs-markup. It seems unlikely that the University was running this version due to the "SMB Name Wildcard" alerts. That alert is not found in Snort version 2.0.1 the most current version it can be found in is version 1.6 released in March of 2000. The "NTPDX" alert is also first found in version 1.6, http://cvs.snort.org/viewcvs.cgi/snort/rules/exploit.rules.diff?r1=1.5&r2=1.6.

A UDP packet with a destination port 123 and a data size greater than

128 bytes would have triggered this alert.

Source Address

The source address "198.64.140.205" could easily have been spoofed for both of the alerts. Both protocols are based on the user datagram protocol (UDP) a connectionless transport protocol in the TCP/IP stack. Individual packets triggered both alerts and there is no correlation of the external IP to any other traffic. If the attack were successful the attacker would have created another communications channel to send commands to the host. This second channel could be outbound or inbound. The logs do not show any other activity on September 16th. The whois database shows the IP registered to a large hosting company Verio.

```
whois -h whois.arin.net 198.64.140.205

OrgName:    Verio, Inc.
OrgID:      VRIO
Address:    8005 South Chester Street
Address:    Suite 200
City:       Englewood
StateProv:  CO
PostalCode: 80112
Country:    US

ReferralServer: rwhois://rwhois.verio.net:4321/

NetRange:   198.63.0.0 - 198.66.255.255
CIDR:       198.63.0.0/16, 198.64.0.0/15, 198.66.0.0/16
NetName:    VRIO-198-063
NetHandle:  NET-198-63-0-0-1
Parent:     NET-198-0-0-0-0
NetType:    Direct Allocation
NameServer: NS0.VERIO.NET
NameServer: NS1.VERIO.NET
NameServer: NS2.VERIO.NET
NameServer: NS3.VERIO.NET
NameServer: NS4.VERIO.NET
Comment:    *Rwhois information on assignments from this block available
Comment:    at rwhois.verio.net port 4321
RegDate:    1993-02-11
Updated:    2005-03-02

TechHandle: VIA4-ORG-ARIN
TechName:   Verio, Inc.
TechPhone:  +1-303-645-1900
TechEmail:  vipar@verio.net

OrgAbuseHandle: VAC5-ARIN
OrgAbuseName:   Verio Abuse Contact
OrgAbusePhone:  +1-800-551-1630
OrgAbuseEmail:  abuse@verio.net

OrgNOCHandle: VSC-ARIN
OrgNOCName:   Verio Support Contact
OrgNOCPhone:  +1-800-551-1630
OrgNOCEmail:  support@verio.net

OrgTechHandle: VIA4-ORG-ARIN
OrgTechName:   Verio, Inc.
OrgTechPhone:  +1-303-645-1900
OrgTechEmail:  vipar@verio.net

# ARIN WHOIS database, last updated 2005-03-13 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.


Found a referral to rwhois.verio.net:4321.

%rwhois V-1.5:0078b6:00 rwhois.verio.net (Vipar 0.1a. Comments to vipar@verio.net)
network:Class-Name:network
network:Auth-Area:198.64.128.0/19
network:ID:NETBLK-W061-198-064-128.127.0.0.1/32
network:Handle:NETBLK-W061-198-064-128
```

```
network:Network-Name:W061-198-064-128
network:IP-Network:198.64.128.0/19
network:In-Addr-Server;I:NS931-HST.127.0.0.1/32
network:In-Addr-Server;I:NS1829-HST.127.0.0.1/32
network:In-Addr-Server;I:NS4208-HST.127.0.0.1/32
network:IP-Network-Block:198.64.128.0 - 198.64.159.255
network:Org-Name:Verio Advanced Hosting - Dulles
network:Street-Address:22451 Shaw Rd
network:City:Sterling
network:State:VA
network:Postal-Code:20166
network:Country-Code:US
network:Tech-Contact;I:IA17312-VRIO.127.0.0.1/32
network:Created:2002-03-13 17:07:41+00
network:Updated:2002-03-13 17:07:41+00
```

The IP addresses in some of the log files are obfuscated but in some of the logs they were not. It was therefore possible to perform a DNS lookup on the destination IP. It resolved to a PPP connection. If the University DNS is to be trusted one can assume this IP address is a dial-in connection. It is very likely part of a dial-in pool. This can explain some of the profile changes noted in the scanning.

Attack Mechanism

The NTPDX exploit attacks the network time protocol service on UDP port 123. It sends an over size datagram to the service and overflows a buffer in the NTP server. There are two common versions of NTP version 3 and version 4. According to the original bugtrac post versions greater than 4.0.99k are not vulnerable, note subject line.

```
To:  BugTraq
Subject:  ntpd =< 4.0.99k remote buffer overflow
Date:  Apr 4 2001 10:27PM
Author:  Przemyslaw Frasunek <venglin freebsd lublin pl>
Message-ID:  <20010404222701.X91913@riget.scene.pl>
```

Further analysis is difficult given the fact that the scan log for September 16th is empty. Also the terminal server logs need to be analyzed to locate the dial-in users connected at the time of the alerts.

It should also be noted that the rule as quoted above could trigger false positives. According to a Neohapsis posting an ntpdc query can trigger the rule by requesting an NTP server return a response that is often greater than 128 bytes: http://archives.neohapsis.com/archives/snort/2002-12/0413.html.

Correlation

The original BugTraq can be found at http://www.securityfocus.com/archive/1/174011 dated April 4th, 2001 and submitted by "Przemyslaw Frasunek."

The original advisory can be found at http://www.securityfocus.com/advisories/3200 dated April 5th, 2001.

An in-depth analysis of the exploit code can be found in Miika Turkia's GCIA practical: http://www.giac.org/certified_professionals/practicals/gcia/0352.php.

The rule first appeared in Snort version 1.6

http://cvs.snort.org/viewcvs.cgi/snort/rules/exploit.rules.diff?r1=1.5&r2=1.6 dated Mon Apr 9 06:39:44 2001 UTC. A recommendation for updating the rule can be found at http://archives.neohapsis.com/archives/snort/2002-12/0413.html posted by James (jamesh_at_cybermesa.com) on Sat Dec 14 2002 - 05:02:29 CST.

The original Request for Comment on NTP is "RFC 958 - Network Time Protocol (NTP)" updated in "RFC 1305 - Network Time Protocol (Version 3) Specification, Implementation." More information can be found at http://www.ntp.org/ntpfaq/NTP-s-def.htm#AEN1438.

Active Targeting

There is no evidence of active targeting. There is no scan activity targeting either NTP or BO. The lack of active targeting coupled with the knowledge that the target is a dial-in greatly increases the probability that these alerts are false positives.

Severity

Severity = (criticality + lethality) – (system countermeasures + network countermeasures)
(1 + 5) – (1 + 2) = 3

Criticality = 1
The machine appears to be a member of a dial-in pool. End-user stations generally connect via dial-in therefore it is almost certainly NOT a server or infrastructure device.
Lethality = 5
The attack would result in the attacker gaining full control of the system.
System Countermeasures = 1
The NTP service is often run as the super user and unlikely to be firewalled, logged, or updated on end user dial-in systems.
Network Countermeasures = 2
A University network providing dial-in network access with a public IP pool is unlikely to firewall the connections. Due to the fact that this paper is being written they are certainly running the traffic through an IDS system.

Recommendations

The system should be located to substantiate the data by an incident handling team. The IDS rules should be tuned to include the change recommended in the Snort Users mailing list post on Neohapsis. Increasing the dgram check size to 188 will filter out possible false positives caused by ntpdc queries. The Back Orifice rule should also be updated to current Snort rules that check datagram content as well as port numbers. This will provide the IDS team with alerts with a lower percentage of false positives.

The University appears to run multiple NTP servers on their network. Given this fact NTP packets should be blocked at the perimeter for all systems except the NTP servers. Internal hosts could then synchronize time off of the

University provided servers and be protected from exploitation from the Internet at large. Host based firewalls should also block NTP traffic on key University systems to only allow connections with the University NTP servers.

Detect 2

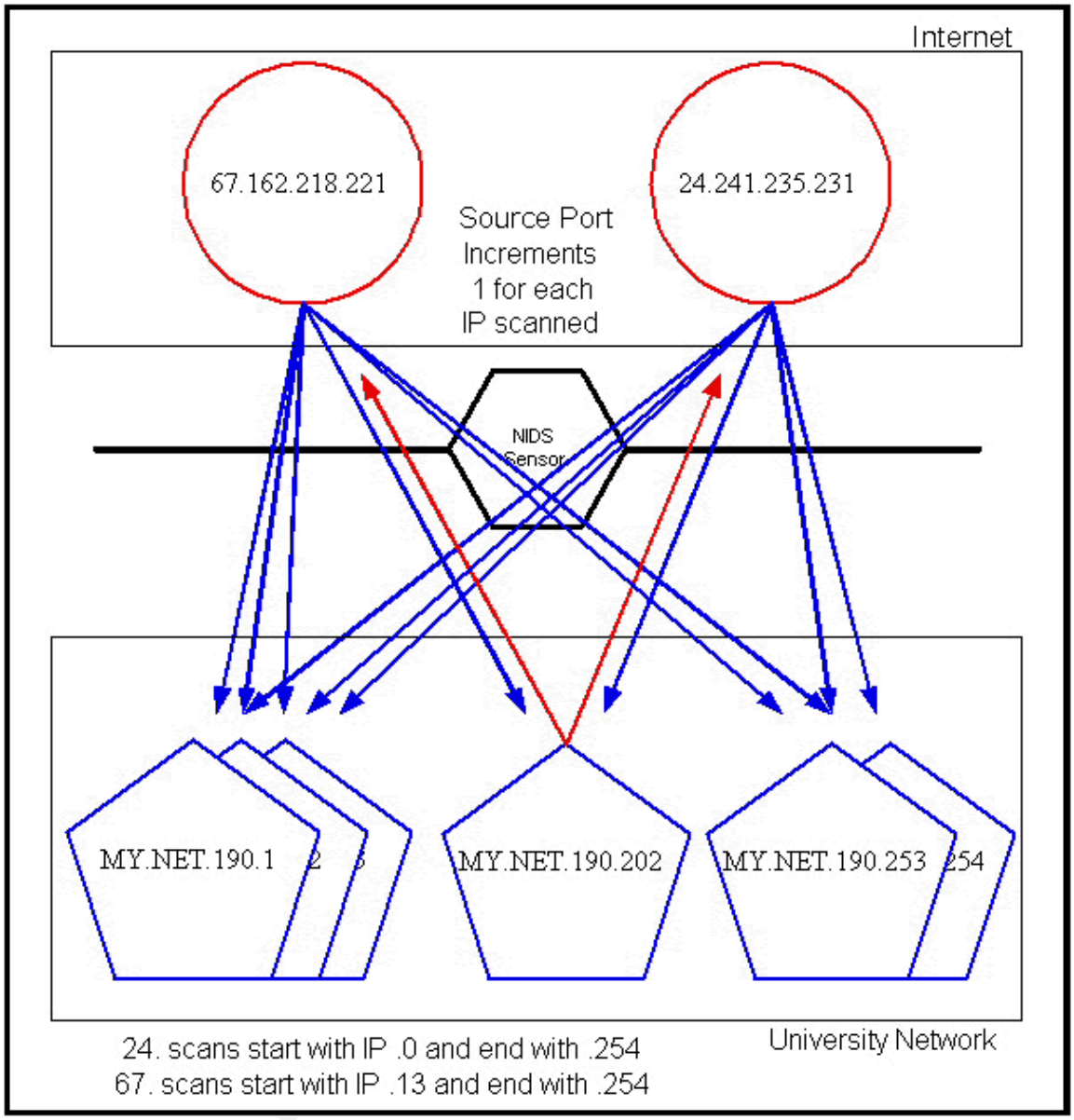
Nine Hundred and eighty two alerts were triggered with the message "Possible trojan server activity." Of these alerts two horizontal scans were conducted against the University's MY.NET.190.0/24 network. The scans were searching for port 27374 a common trojan server port. Currently http://www.treachery.net/tools/ports/ lists 14 different trojans that use this port. Security Focus lists an advisory for Subseven version 1.9 as early as 1999 indicating a number of versions of Subseven would have been in circulation at the time of these alerts. The site also contains advisories regarding the Ramen worm as early as 2001 indicating that multiple malware programs were using port 27374 at the time these alerts were triggered.

The log entries clearly show the horizontal scans for port 27374, in both the alert log and the scan log. More troubling is the fact that the alert logs show responses indicating that some of the hosts scanned responded to the scans.

```
ALERT LOG for 24.241.235.231 to port 27374
09-19 06:38:54 Possible trojan server activity 24.241.235.231  2048 257.257.6.15    27374
09-19 06:38:55 Possible trojan server activity 24.241.235.231  2048 257.257.6.15    27374
09-19 06:38:56 Possible trojan server activity 24.241.235.231  2048 257.257.6.15    27374
09-19 06:43:19 Possible trojan server activity 24.241.235.231  4160 257.257.16.90   27374
09-19 06:43:24 Possible trojan server activity 24.241.235.231  4467 257.257.16.114  27374
09-19 07:58:13 Possible trojan server activity 24.241.235.231  1201 257.257.190.0   27374
09-19 07:58:13 Possible trojan server activity 24.241.235.231  1202 257.257.190.1   27374
09-19 07:58:13 Possible trojan server activity 24.241.235.231  1203 257.257.190.2   27374
09-19 07:58:13 Possible trojan server activity 24.241.235.231  1204 257.257.190.3   27374
09-19 07:58:13 Possible trojan server activity 24.241.235.231  1205 257.257.190.4   27374
~Many incrementing source ports and destination IPs in between~
09-19 07:58:38 Possible trojan server activity 24.241.235.231  3100 257.257.190.236 27374
09-19 07:58:38 Possible trojan server activity 24.241.235.231  3101 257.257.190.237 27374
09-19 07:58:38 Possible trojan server activity 24.241.235.231  3110 257.257.190.246 27374
09-19 07:58:38 Possible trojan server activity 24.241.235.231  3109 257.257.190.245 27374
09-19 07:58:38 Possible trojan server activity 24.241.235.231  3111 257.257.190.247 27374
09-19 07:58:40 Possible trojan server activity 24.241.235.231  3255 257.257.190.251 27374
09-19 07:58:40 Possible trojan server activity 24.241.235.231  3256 257.257.190.252 27374
09-19 07:58:40 Possible trojan server activity 24.241.235.231  3257 257.257.190.253 27374
09-19 07:58:40 Possible trojan server activity 24.241.235.231  3258 257.257.190.254 27374

Responding stations
09-19 06:38:54 Possible trojan server activity 257.257.6.15    27374 24.241.235.231  2048
09-19 06:38:55 Possible trojan server activity 257.257.6.15    27374 24.241.235.231  2048
09-19 06:38:56 Possible trojan server activity 257.257.6.15    27374 24.241.235.231  2048
09-19 07:58:13 Possible trojan server activity 257.257.190.1   27374 24.241.235.231  1202
09-19 07:58:14 Possible trojan server activity 257.257.190.1   27374 24.241.235.231  1202
09-19 07:58:14 Possible trojan server activity 257.257.190.1   27374 24.241.235.231  1202
09-19 07:58:33 Possible trojan server activity 257.257.190.202 27374 24.241.235.231  2806
09-19 07:58:33 Possible trojan server activity 257.257.190.203 27374 24.241.235.231  2807
09-19 07:58:34 Possible trojan server activity 257.257.190.203 27374 24.241.235.231  2807

Entries in the scan log
2003-Sep-19 07:58:14 24.241.235.231  1202    .190.1    27374   TCP     SYN ******S*
2003-Sep-19 07:58:15 24.241.235.231  1350    .190.8    27374   TCP     SYN ******S*
2003-Sep-19 07:58:15 24.241.235.231  1351    .190.9    27374   TCP     SYN ******S*
2003-Sep-19 07:58:15 24.241.235.231  1352    .190.10   27374   TCP     SYN ******S*
2003-Sep-19 07:58:15 24.241.235.231  1353    .190.11   27374   TCP     SYN ******S*
```

```
2003-Sep-19 07:58:15 24.241.235.231  1354    .190.12   27374   TCP     SYN ******S*
~Many incrementing source ports and destination IPs in between~
2003-Sep-19 07:58:38 24.241.235.231  3100    .190.236  27374   TCP     SYN ******S*
2003-Sep-19 07:58:38 24.241.235.231  3101    .190.237  27374   TCP     SYN ******S*
2003-Sep-19 07:58:38 24.241.235.231  3110    .190.246  27374   TCP     SYN ******S*
2003-Sep-19 07:58:38 24.241.235.231  3109    .190.245  27374   TCP     SYN ******S*
2003-Sep-19 07:58:38 24.241.235.231  3111    .190.247  27374   TCP     SYN ******S*
2003-Sep-19 07:58:40 24.241.235.231  3255    .190.251  27374   TCP     SYN ******S*
2003-Sep-19 07:58:40 24.241.235.231  3256    .190.252  27374   TCP     SYN ******S*
2003-Sep-19 07:58:40 24.241.235.231  3257    .190.253  27374   TCP     SYN ******S*
2003-Sep-19 07:58:40 24.241.235.231  3258    .190.254  27374   TCP     SYN ******S*

ALERT LOG for 67.162.218.221 to port 27374
09-19 21:05:58 Possible trojan server activity 67.162.218.221  3529 257.257.16.90   27374
09-19 21:05:59 Possible trojan server activity 67.162.218.221  3752 257.257.16.106  27374
09-19 21:05:59 Possible trojan server activity 67.162.218.221  3760 257.257.16.114  27374
09-19 21:43:40 Possible trojan server activity 67.162.218.221  3898 257.257.190.13  27374
09-19 21:43:40 Possible trojan server activity 67.162.218.221  3899 257.257.190.14  27374
09-19 21:43:40 Possible trojan server activity 67.162.218.221  3900 257.257.190.15  27374
~Many incrementing source ports and destination IPs in between~
09-19 21:43:51 Possible trojan server activity 67.162.218.221 3694 257.257.190.250 27374
09-19 21:43:51 Possible trojan server activity 67.162.218.221 3693 257.257.190.249 27374
09-19 21:43:51 Possible trojan server activity 67.162.218.221  3695 257.257.190.251 27374
09-19 21:43:51 Possible trojan server activity 67.162.218.221  3696 257.257.190.252 27374
09-19 21:43:52 Possible trojan server activity 67.162.218.221  4050 257.257.190.254 27374

Responding stations
09-19 21:43:44 Possible trojan server activity 257.257.190.100 27374 67.162.218.221  1543
09-19 21:43:49 Possible trojan server activity 257.257.190.203 27374 67.162.218.221  3089
09-19 21:43:49 Possible trojan server activity 257.257.190.202 27374 67.162.218.221  3088
09-19 21:43:49 Possible trojan server activity 257.257.190.202 27374 67.162.218.221  3088
09-19 21:43:49 Possible trojan server activity 257.257.190.203 27374 67.162.218.221  3089
09-19 21:43:50 Possible trojan server activity 257.257.190.203 27374 67.162.218.221  3089
09-19 21:43:50 Possible trojan server activity 257.257.190.202 27374 67.162.218.221  3088

Entries in the scan log
2003-Sep-19 21:43:40 67.162.218.221  3898    130.85.190.13   27374   TCP     SYN ******S*
2003-Sep-19 21:43:40 67.162.218.221  3899    .190.14   27374   TCP     SYN ******S*
2003-Sep-19 21:43:40 67.162.218.221  3900    .190.15   27374   TCP     SYN ******S*
2003-Sep-19 21:43:40 67.162.218.221  3902    .190.16   27374   TCP     SYN ******S*
~Many incrementing source ports and destination IPs in between~
2003-Sep-19 21:43:51 67.162.218.221  3694    .190.250  27374   TCP     SYN ******S*
2003-Sep-19 21:43:51 67.162.218.221  3693    .190.249  27374   TCP     SYN ******S*
2003-Sep-19 21:43:51 67.162.218.221  3695    .190.251  27374   TCP     SYN ******S*
2003-Sep-19 21:43:51 67.162.218.221  3696    .190.252  27374   TCP     SYN ******S*
2003-Sep-19 21:43:52 67.162.218.221  4050    .190.254  27374   TCP     SYN ******S*
```

A link graph clearly illustrates the one to many relationship of the scan by each of the offending hosts. For each attacker the source ports increment by one for each IP scanned. The host MY.NET.190.202 is taken from the group to represent the few machines that responded to the scan.



Detect Generation

A rule was not found in any Snort distribution containing the message text "Possible trojan server activity" indicating this is likely a custom rule. Some similar rules were found that will be used as a basis for comparison with this traffic.

```
Snort version 1.6 rule with port 27374
alert tcp $EXTERNAL_NET 27374 -> $HOME_NET any (msg:"BACKDOOR SIG - SubSseven 22"; flags:
A+; content: "|0d0a5b52504c5d3030320d0a|"; reference:arachnids,485; sid:103; rev:1;)

Rules from Packet Storm Security dated 06/08/2000
alert tcp $HOME_NET 27374 -> !$HOME_NET any (msg:"IDS279 - BACKDOOR ACTIVITY-Possible
Subseven v2.1"; flags:SA;)
alert tcp !$HOME_NET any -> $HOME_NET 27374 (msg:"IDS279 - BACKDOOR ATTEMPT-Subseven
v2.1"; flags:S;)
```

The first rule specifies a content match on the packet that would make it far to specific to warrant the message in the alerts. The second and third examples are closer to what would be expected for a general rule. They match on a source or destination port and look for TCP and synack and syn flags to be set. The match for the syn and ack flags are important because if they were not included then traffic to a closed port would generate two alerts (such as we see in our logs for MY.NET.190.202) because the station would respond with a packet with the rst flag set. As the rules are they match our traffic very well indicating a stimulus to the port and a response from the host further ensuring a valid alert.

Source Address

The source addresses of the attack both trace back to cable Internet providers. One provider located in Wisconsin and one located in Michigan. The source IP address could have been spoofed if our rules check simply for the syn and syn ack and do not verify a completed TCP stream. The original bugtraq for using a zombie host for TCP idle scanning with the IP ID field was posted well before these alerts were generated. Antirez posted an example of the method to Bugtraq in late 1998.

Attack Mechanism

The scans for port 27374 are not actively attacking the hosts on the .190.0/24 network. Instead they are probing to find machines that have already been infected. It is probable that the delivery of a trojan (Subseven for example) was attempted earlier by some unrelated means (email for example). These scans would then be done as a follow-up checking for successful deployment of the original attack.

Correlation

The example rules used can be found hosted by Packet Storm at the following URL http://packetstormsecurity.nl/sniffers/snort/06082kbackdoor.rules.

A quick explanation of the Ramen worm and its use of port 2734 can be found here http://www.sans.org/y2k/ramen.htm.

An early advisory referencing the Subseven trojan is located here http://www.securityfocus.com/advisories/1644.

Some early posts showing rules to detect Ramen and Subseven using snort can be found here http://www.securityfocus.com/archive/96/191589.

The earliest reference to using the IP ID idle scanning technique is described here http://www.kyuzz.org/antirez/papers/dumbscan.html.

An excellent walkthrough of filtering false positive alerts out of the results for the “Possible trojan” rule can be found here http://www.lurhq.com/idsindepth.html.

Active Targeting

These traffic patterns are a perfect example of active targeting. These two IP sources (or the source that is spoofing these addresses) is actively scanning

through nearly 256 IP addresses attempting to locate machines that have been compromised by trojan horse programs.

Severity

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

Criticality = 3
A reverse DNS lookup was performed on the IP addresses that responded during the scan. Each IP address contains a unique name; two of them contain the string “VPN” indicating they may be virtual private network (VPN) gateway hosts. The hosts are therefore considered to be someone critical systems.
Lethality = 2
Unless accompanied by an original attack to place the trojan program on the hosts the scan would have no effect.
System Countermeasures = 2
No information regarding the hosts was provided. Since a few of the systems responded, it is assumed that few system countermeasures are in place.
Network Countermeasures = 1
Given that multiple machines responded to the scan either there is no firewall in place or it is not blocking traffic to these hosts or ports.

Recommendations

The hosts that responded to the scan need to be inspected by an incident response team. There is a high probability that these alerts do in fact indicate malicious traffic and the hosts are likely running a backdoor program allowing full remote control of the systems.

A firewall rule should be created blocking all inbound traffic to port 27374. There is no legitimate service identified as using that port therefore it should only be allowed into the network as a response to established network connections. Any outgoing traffic destined for port 27374 should be logged as it is likely traffic from an internal host scanning for Internet hosts infected with a trojan.

Detect 3

The last analysis started with an investigation into the “SUNRPC highport access!” and the “Attempted Sun RPC high port access” alerts. A large number of the alerts were sourced from the same IP address and port, “64.12.28.110:5190”. America Online Instant Messenger (AIM) uses Port 5190 and the IP address is registered to AOL. With this traffic removed another IP address stands out, “63.250.195.10” belonging to Yahoo.

```
AIM Traffic
09-15 11:32:05 SUNRPC highport access! 64.12.28.110    5190    257.257.70.38   32771


09-19 17:05:56 SUNRPC highport access! 65.165.91.219   80      257.257.97.96   32771
```

```
09-19 17:05:59 SUNRPC highport access! 65.165.91.219   80      257.257.97.96   32771

09-15 13:34:26 Attempted Sun RPC high port access 63.250.195.10 44553 .153.153 32771
09-18 00:43:29 Attempted Sun RPC high port access 63.250.195.10 59530 .69.208  32771

09-15 13:34:26 Attempted Sun RPC high port access 63.250.195.10 44553 .153.153 32771
09-15 13:28:15 High port 65535 udp - possible Red Worm - traffic 63.250.195.10 53247
257.257.153.153 65535
09-17 13:35:18 High port 65535 udp - possible Red Worm - traffic 63.250.195.10 65535
257.257.153.153 65488
```

The Yahoo IP was found attacking with a number of different signatures mostly targeting *IX based hosts. The NTPDX exploit was listed earlier, Sun RPC high port refers to UDP port 32771 an RPC port running the rusers daemon, Red Worm was somewhat of a copy cat follow-up to the Ramen worm attacking Linux, and TFTP (more cross platform) is a file transfer protocol that lacks authentication.

```
Attempted Sun RPC high port access
EXPLOIT NTPDX buffer overflow
High port 65535 udp - possible Red Worm - traffic
TFTP - External UDP connection to internal tftp server
TFTP - Internal UDP connection to external tftp server
```

The Yahoo IP also attacked a number of targets. The IP addresses were resolved to names in search of some commonality. Of the addresses that resolved, a number of them appear as public library PCs. This would suggest Windows clients that would not be vulnerable to the alerts listed.

```
121.150..in-addr.arpa domain name pointer serial11.lib..edu.
85.150..in-addr.arpa domain name pointer tec18.lib..edu.
157.152..in-addr.arpa domain name pointer lib037pc01.ucslab..edu.
169.152..in-addr.arpa domain name pointer lib037pc13.ucslab..edu.
250.152..in-addr.arpa domain name pointer lib037pc57.ucslab..edu.
153.153..in-addr.arpa domain name pointer libstkpc13.libpub..edu.
31.153..in-addr.arpa domain name pointer refweb02.libpub..edu.
36.153..in-addr.arpa domain name pointer refweb07.libpub..edu.
87.153..in-addr.arpa domain name pointer refweb23.libpub..edu.
159.53..in-addr.arpa domain name pointer ecs333pc29.ucslab..edu.
196.53..in-addr.arpa domain name pointer ecs020pc01.ucslab..edu.
215.53..in-addr.arpa domain name pointer ecs104pc15.ucslab..edu.
208.69..in-addr.arpa domain name pointer lib-69-208.pooled..edu.
197.70..in-addr.arpa domain name pointer ecs020pc02.ucs..edu.
61.97..in-addr.arpa domain name pointer ppp1-61.dialup..EDU.
```

The scan log also contained a large number of hits from the Yahoo IP, 43,050 all were UDP. Of these 11,335 different ports were hit. There appeared to be no pattern among the source ports or the destination ports, they appeared random.

The only non-hostile explanation for the traffic was found through Google searches. A Mcafee firewall forum posting was found containing the IP address. The user was having problem using Microsoft Media Player and it was triggering alerts to his firewall. Sites were also found that contained URLs pointing to the Yahoo IP with the mms:// protocol. Two documents on Microsoft's website were found describing the Microsoft Media Server and its associated protocols. The

following excerpt from the site describes what appears to be the traffic reported in the alerts.

In a TCP connection there is only one socket created. (A socket is an identifier for a particular service on a particular node on a network.) You therefore need only one port number on the client and one on the server. Commands (such as play, pause, and fast forward) and data (audio and video) are sent across the same socket connection. In UDP connections, however, the client makes a TCP connection to the server and sends commands over it. The server then opens a UDP socket to the client. It is over this second socket that the audio and video data is sent and it is this second socket that firewalls and proxies typically block.

Detect Generation

```
alert udp any any -> $HOME_NET 32771 (msg: "Attempted Sun RPC high port access";)
```

This rule generated the original alert that began this analysis. The NTPDX alert was covered earlier and the rest of the alerts were generated by apparent custom rules. This rule checks for UDP packets destined for port 32771, the port associated with the rusers RPC daemon on Sun hosts. The rule does not however provide any content matching, thus the rule can easily trigger when client programs bind the port as a random source port. The rules alerting for TFTP traffic may be similar. They would be less likely to be triggered however because they match for port 69. Ports below 1024 are typically reserved for servers and thus most clients pick random ports starting above 1024. An example of this is listed in the correlation section regarding the Microsoft SQL client.

Source Address

All the packets associated with these detects were transmitted using UDP. UDP is connectionless meaning a single packet can be sent without any need for a response. It is also unlikely that any of the rules triggered did any content checking; meaning a single packet of any type could have triggered these rules. The only factor that points to a valid source address is the shear volume of traffic and the probable distribution across similar end stations. If this traffic is indeed Windows Media Player traffic then the source address is valid. The only remaining suspicion against this theory is the fact that ports from 0-1024 are logged, it seems unlikely that low numbered ports would be included in any applications random source/destination port pool.

Attack Mechanism

The Sun RPC alert is triggered to alert administrators that the rusersd RPC program is being probed. This service should be disabled per standard Solaris hardening procedures. It gives a remote user a list of local accounts to assist in further attacks against a host. The TFTP alerts notify administrators that files are being transferred into or out of the network using a protocol that has no authentication. TFTP is typically utilized to backup configurations or to upload firmware to network infrastructure devices. It should not be used for Internetworking transfers.

The other possible explanation for these traffic patterns taken as a whole

is that the Yahoo IP is port scanning a number of University hosts. These scans may be spoofed to obscure true attacks/scans or the machine may be compromised. The fact that Yahoo is the owner of the address space is positive, as they should prove to be helpful in further investigations of the traffic.

```
OrgName:    Yahoo! Broadcast Services, Inc.
OrgID:      YAHO
Address:    701 First Avenue
City:       Sunnyvale
StateProv:  CA
PostalCode: 94089
Country:    US

NetRange:   63.250.192.0 - 63.250.223.255
CIDR:       63.250.192.0/19
NetName:    NETBLK2-YAHOOBS
NetHandle:  NET-63-250-192-0-1
Parent:     NET-63-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.YAHOO.COM
NameServer: NS2.YAHOO.COM
NameServer: NS3.YAHOO.COM
NameServer: NS4.YAHOO.COM
NameServer: NS5.YAHOO.COM
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    1999-11-24
Updated:    2003-05-06

TechHandle: NA258-ARIN
TechName:   Netblock Admin
TechPhone:  +1-408-349-3300
TechEmail:  netblockadmin@yahoo-inc.com

OrgTechHandle: NA258-ARIN
OrgTechName:   Netblock Admin
OrgTechPhone:  +1-408-349-3300
OrgTechEmail:  netblockadmin@yahoo-inc.com

# ARIN WHOIS database, last updated 2005-03-20 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

Correlation

F-Secure's site was used to research the Red Worm attack http://www.f-secure.com/v-descs/adore.shtml.

Microsoft describes the behaviour of its SQL client choosing a random source port in this article http://support.microsoft.com/kb/q287932/.

The reference to Microsoft Media Server can be found at these two URLs: http://www.microsoft.com/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/Default.asp?url=/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/wmserver/wmsmmsservercontrolprotocol.asp and http://www.microsoft.com/technet/prodtechnol/Windows2000Pro/maintain/w2kmngd/14_2kwmp.mspx.

The McAfee firewall forum posting referencing the Yahoo IP can be found here http://forums.mcafeehelp.com/viewtopic.php?t=8978.

A lookup using http://www.treachery.net/tools/ports/lookup.cgi port lookup tool correlated port 5190 to AIM.

A security focus article discusses the rusersd service and lists it as a

standard service to disable during hardening http://www.securityfocus.com/infocus/1365.

Active Targeting

The large number of ports utilized across numerous hosts indicates either active targeting or a client application designed to drive security analysis crazy. If it is port scanning then the attacker will likely know if the targets are firewalled. If they are not firewalled then the attacker will know which ports are open and based on this information the platform of the system will be evident.

Severity

Severity = (criticality + lethality) – (system countermeasures + network countermeasures)
(1 + 5) – (1 + 1) = 4

Criticality = 1
None of the systems targeted appear particularly critical. They all have generic hostnames referencing a number and many contain “pc” indicating an end users station. While downtime is not enjoyable on any stations these appear to be the most expendable.
Lethality = 5
Only the NTPDX exploit would have resulted in super user privileges. The Sun RPC alerts would have only divulged valid usernames and the TFTP traffic would indicate someone accessing a TFTP server. That server should however not be accessible outside the network and may contain sensitive data. The most lethal would be the Red Worm alerts. They may indicate that a successful super user compromise had already occurred.
System Countermeasures = 1
The end user station identified would be the least protected systems on a University network. They are unlikely to receive regular attention regarding patch installation and almost certainly contain no host-based firewall as that may interfere with their use.
Network Countermeasures = 1
Just as they are unlikely to use host-based firewalling they are also unlikely to utilize a network-based firewall. The only positive is that these logs were collected meaning their traffic is monitored by a network based intrusion detection system.

Recommendations

Yahoo should be contacted to verify the host that originated this traffic. If it is valid traffic they will likely be able to provide details regarding the protocol in question. If they are unaware of a media client exhibiting these characteristics then they will want to verify whether the traffic originated on their network and verify the integrity of the host machine. If the traffic is valid then Windows Media Player can be configured to stream the traffic over HTTP. This would make this traffic much easier to identify in the future.

The IDS system should be configured to provide "content" matches whenever possible. Content matches check the data inside of the packet to verify that the payload matches a signature before triggering an alert. These types of rules provide a much lower level of false positives thereby increasing the probability that analysis will locate malicious traffic in the IDS logs.

Network Statistics

Alert File Statistics

Number of Alerts: 747578
Unique alerts and counts can be found under "Detect Overview."
Unique alerts: 49
Unique source IP addresses for alerts: 1667
Unique University source IP addresses for alerts: 159
Unique Internet source IP addresses for alerts: 1508
Top five University source IP addresses and alert count:
579678 257.257.162.118
48963 257.257.150.220
14218 257.257.150.133
3868 257.257.70.207
1933 257.257.21.50
Top five Internet source IP addresses and alert count:
579678 257.257.162.118
48963 257.257.150.220
14218 257.257.150.133
3868 257.257.70.207
1933 257.257.21.50
Unique destination IP addresses for alerts: 591949
Unique University destination IP addresses for alerts: 693
Unique Internet destination IP addresses for alerts: 591256
Top five University destination IP addresses and alert count:
57051 257.257.30.4
9657 257.257.30.3
5827 257.257.70.207
2323 257.257.24.15
1577 257.257.70.38
Top five Internet destination IP addresses and alert count:
3864 216.231.173.92
2923 24.211.131.102
2671 67.69.108.230
2617 205.134.166.194
2337 209.126.216.244
Unique source ports for alerts: 3625
Unique University source ports for alerts: 585
Unique Internet source ports for alerts: 3224
Top five University source ports and alert count:
578984 1025
14968 4253
14584 99999 <- No SRC PORT SPECIFIED IN ALERT

10418 4018
9500 1184
Top five Internet source ports and alert count:
17141 3738
15764 3488
6292 65535
3447 1247
2664 1091
Unique destination ports for alerts: 1035
Unique University destination ports for alerts: 935
Unique Internet destination ports for alerts: 143
Top five University destination ports and alert count:
48970 51443
11239 524
5824 12203
4217 80
2340 3019
Top five Internet destination ports and alert count:
643860 137
16608 99999 <- No DST PORT SPECIFIED IN ALERT
4211 65535
230 27374
190 80
Top five University source IP addresses and count from portscan entries in alert file:
22821 257.257.84.194
5064 257.257.70.53
3560 257.257.163.235
2297 257.257.80.243
1796 257.257.1.5
Top five Internet source IP addresses and count from portscan entries in alert file:
3639 148.63.83.0
1510 66.117.22.125
1112 194.249.91.190
1014 66.250.55.114
542 69.1.65.188

Scan File Statistics

Top five University source IP addresses and count:
2597670 130.85.1.3
1521633 130.85.84.194
1315532 130.85.70.53
580124 130.85.1.5
356322 130.85.80.243
Top five destination ports/protocols and destination IP/ports for top five University source IP addresses filtered for ports 135 137 25 and 53:

```
for i in $(echo 'select srcip from scan where srcip LIKE "MY.NET.%" and dstport != "53"
and dstport != "135" and dstport != "25" and dstport != "137" order by srcip;' | mysql
gcia | uniq -c | sort -nr | head -n5 | cut -c9-100); do
 echo
 echo $i
```

```
 echo "select dstport,prot from scan where srcip LIKE '$i' order by dstport,prot;" | mysql
gcia | uniq -c | sort -nr | head -n5
 echo "select dstip,dstport,prot from scan where srcip LIKE '$i' order by
dstip,dstport,prot;" | mysql gcia | uniq -c | sort -nr | head -n5;
done

130.85.70.53
1225687 135     TCP
  89844 80      TCP
      1 dstport prot
      1 443     TCP
     40 130.84.173.131  135     TCP
     38 130.84.111.115  135     TCP
     38 130.84.110.173  135     TCP
     37 130.84.44.120   135     TCP
     37 130.84.228.44   135     TCP

130.85.70.207
   2401 33774   UDP
   2056 65090   UDP
   1997 18304   UDP
   1978 11564   UDP
   1808 33513   UDP
   2256 24.224.188.141  33774   UDP
   2056 12.222.24.242   65090   UDP
   1997 69.27.66.247    18304   UDP
   1978 142.217.110.70  11564   UDP
   1808 81.152.67.233   33513   UDP

130.85.84.210
  26394 6257    UDP
    426 80      TCP
    149 16257   UDP
     83 6256    UDP
     77 26257   UDP
     98 172.133.50.228  6257    UDP
     90 62.211.252.6    6257    UDP
     83 68.72.171.140   6257    UDP
     78 68.14.89.26     6257    UDP
     69 80.116.225.5    6257    UDP

130.85.98.132
  12101 22321   UDP
   4289 7674    UDP
     14 80      TCP
      5 1025    UDP
      4 2865    UDP
      9 220.122.198.101 22321   UDP
      8 211.212.96.4    22321   UDP
      7 220.91.73.39    22321   UDP
      7 220.81.215.195  22321   UDP
      7 219.248.155.236 22321   UDP

130.85.82.2
   1626 4046    UDP
   1621 32801   UDP
   1605 2645    UDP
   1594 1173    UDP
   1553 1100    UDP
```

```
   1626 68.34.149.132   4046    UDP
   1621 4.3.172.73      32801   UDP
   1605 24.209.234.234  2645    UDP
   1594 24.197.242.66   1173    UDP
   1553 24.79.219.29    1100    UDP
```

Top five Internet source IP addresses and count:

```
 43050 63.250.195.10
 25952 217.58.104.215
 23234 168.215.234.10
 22965 217.236.57.37
 22360 24.220.60.195
```

Top five destination ports/protocols and destination IP/ports for top five Internet source IP addresses filtered for port 80:

```
for i in $(echo 'select srcip from scan where srcip NOT LIKE "130.85.%" and dstport !=
"80" order by srcip;' | mysql gcia | uniq -c | sort -nr | head -n5 | cut -c9-100); do
 echo
 echo $i
 echo "select dstport,prot from scan where srcip LIKE '$i' order by dstport,prot;" | mysql
gcia | uniq -c | sort -nr | head -n5
 echo "select dstip,dstport,prot from scan where srcip LIKE '$i' order by
dstip,dstport,prot;" | mysql gcia | uniq -c | sort -nr | head -n5;
done

63.250.195.10
  15679 0       UDP
    750 4046    UDP
    229 1431    UDP
    222 1833    UDP
    215 2902    UDP
   5198 130.85.150.85   0       UDP
   4549 130.85.153.31   0       UDP
   1631 130.85.152.157  0       UDP
   1052 130.85.70.197   0       UDP
   1035 130.85.153.36   0       UDP

211.38.231.66
  21874 34816   TCP
      1 dstport prot
      2 130.85.99.99    34816   TCP
      2 130.85.99.98    34816   TCP
      2 130.85.99.84    34816   TCP
      2 130.85.99.83    34816   TCP
      2 130.85.99.82    34816   TCP

80.143.44.248
  20719 34816   TCP
      1 dstport prot
      3 130.85.21.90    34816   TCP
      2 130.85.99.9     34816   TCP
      2 130.85.99.8     34816   TCP
      2 130.85.99.79    34816   TCP
      2 130.85.99.78    34816   TCP

203.242.231.1
  20432 554     TCP
      1 dstport prot
```

```
      1 38473   UDP
      1 38361   UDP
      2 130.85.99.94    554     TCP
      2 130.85.99.93    554     TCP
      2 130.85.99.92    554     TCP
      2 130.85.99.91    554     TCP
      2 130.85.99.88    554     TCP

213.190.156.194
  20124 34816   TCP
      1 dstport prot
      2 130.85.99.99    34816   TCP
      2 130.85.99.98    34816   TCP
      2 130.85.99.90    34816   TCP
      2 130.85.99.9     34816   TCP
      2 130.85.99.89    34816   TCP
```

Out of Specification File Statistics

```
Datagram length 40: 341
Datagram length 44: 381
Datagram length 48: 136
Datagram length 52: 63
Datagram length 56: 33
Datagram length 60: 13589
Datagram length 64: 177
Datagram matches kazaa: 623
Interesting: 10
```

Analysis Process

The following documents the analysis that was performed upon the log files. It includes the commands and procedures performed on the files and is intended to provide a simple walkthrough.

Analysis Station

The log files have been processed using the following software and hardware:

- Gentoo GNU/Linux Kernel 2.6.10
- AMD Athlon(tm) XP 2400+
- 1GB RAM
- RAID1 set of 80GB IDE HD
- 3ware Inc 3ware PATA RAID Controller
- bash, version 2.05b.0(1)-release
- mysql 4.0.22
- perl, v5.8.4
- nmap version 3.75
- grep (GNU grep) 2.5.1
- coreutils (cut, wc) 5.2.1
- bind-tools (host, dig) 9.2.3
- whois 4.6.11
- vim - Vi IMproved 6.3

Walkthrough

The first task is organizing and unzipping the files. The following three command loops move each file to a directory based on their type (oos, scan, alert) and unzips them.

```
mkdir oos
mkdir alert
mkdir scans
for f in $(ls OOS*); do cp $f oos; gunzip oos/$f; done
for f in $(ls alert.03*); do cp $f alert; gunzip alert/$f; done
for f in $(ls scans.03*); do cp $f scans; gunzip scans/$f; done
```

Alert Logs

We start by analyzing the alert files. First they are concatenated together. The correct format for an alert line is:

```
09/15-00:00:01.263804  [**] SMB Name Wildcard [**] MY.NET.162.118:1025 -> 204.127.165.224:137
```

The grep '^:' finds any lines that start with a ":" and the "-v" switch tells it to exclude those lines. The output of the command is redirected ">" to the alert.new file and a line count "wc –l" is performed to ensure the file contains the approximate number of lines that it should. Doing this removes invalid lines that start with a ":" from our alert file.

```
cd alert
for f in $(ls); do cat $f >> alert; done

grep '^:' -v alert > alert.new
wc -l alert
1514130 alert

wc -l alert.new
1513161 alert.new

rm alert
mv alert.new alert
```

Next the “MY.NET.” entries are replaced with “257.257.” this allows easier filtering because the IP address should only contain numerals. The commands are run four times because there can be multiple instances of “MY.NET” on each line. Also some lines were later found to be missing new line characters at the end thus causing two lines to appear at this point as one. Each time the command is run a line count is performed to ensure we have not lost any alerts. Finally we check for any occurrences of “MY.NET” by searching with grep.

```
cat alert|sed -e 's/MY.NET./257.257./' > alert.new
wc -l alert.new
1513161 alert.new

mv alert.new alert
cat alert|sed -e 's/MY.NET./257.257./' > alert.new
wc -l alert.new
1513161 alert.new

mv alert.new alert
cat alert|sed -e 's/MY.NET./257.257./' > alert.new
wc -l alert.new
1513161 alert.new

mv alert.new alert
cat alert|sed -e 's/MY.NET./257.257./' > alert.new
wc -l alert.new
1513161 alert.new

mv alert.new alert

grep MY.NET alert | wc -l
0
```

There were a number of lines (961) that were missing proper new line characters between them but were otherwise undamaged. Eight lines were also found that contained the content from three alerts. The following commands were run and then repeated to break these lines up properly. Grep is used to find the lines by matching four “[**]” sequences in a single line. Sed is then used to search for the date string preceded by a single character and replace it with the single character a new line and then the date string. Again line counts are performed to ensure we don’t lose data.

```
grep '\[\*\*\].*\[\*\*\].*\[\*\*\]' alert | wc -l
961

wc -l alert
1513189 alert
```

```
sed -e 's#\(.\)\(09/1[56789]-[0-9][0-9]:\)#\1\n\2#' alert >alert.new
wc -l alert.new
1514150 alert.new
echo $[ `cat alert | wc -l` + `grep '\[\*\*\].*\[\*\*\].*\[\*\*\]' alert | wc -l` ]
1514150

grep '\[\*\*\].*\[\*\*\].*\[\*\*\]' alert.new | wc -l
8

mv alert.new alert
```

A number of alerts were found referencing three IP addresses such as "68.54.168.204:3488 -> 257.257.30.4:3488 -> 257.257.30.4:51443". Some lines contained two external addresses followed by an internal. Others contained two internal addresses preceded by an external address. The sed command was used to search for multiple "->" sequences and remove the IP address in between. A line count is performed on a grep to locate the number of corrupted lines. A line count is then performed a diff command of the new and old file to find the number of lines affected by the sed command. Finally because the diff output displays four lines for every line that is different a simple match command is performed to validate the output.

```
$ sed -e 's/->.*->/->/' alert > alert.new

$ wc -l alert
1514115 alert

$ wc -l alert.new
1514115 alert.new

$ grep '..->.*->' alert | wc -l
149

$ grep '..->.*->' alert.new | wc -l
0

$ diff alert alert.new | wc -l
596

$ echo $[ 149 * 4 ]
596

mv alert.new alert
```

Now the Mysql database is created. The schema used is based off of the column names used in the perl script contained in Jason Lam's GCIA Practical. The perl script used later in this document is a modification of that used by Mr. Lam. The perl script contains multiple lines that are commented out, these were used to test the script to ensure the pattern matches were working correctly.

```
mysqladmin -u root -p  create gcia

$ cat mysql.schema
DROP TABLE IF EXISTS portscan;
CREATE TABLE portscan (
  id int(11) DEFAULT '0' NOT NULL auto_increment,
  timeof varchar(24) DEFAULT '' NOT NULL,
```

```
  srcip varchar(16) DEFAULT '' NOT NULL,
  nohost varchar(16) DEFAULT '' NOT NULL,
  tcpno varchar(16) DEFAULT '' NOT NULL,
  udpno varchar(16) DEFAULT '' NOT NULL,
  PRIMARY KEY (id)
);

DROP TABLE IF EXISTS alert;
CREATE TABLE alert (
  id int(11) DEFAULT '0' NOT NULL auto_increment,
  timeof varchar(24) DEFAULT '' NOT NULL,
  action varchar(100) DEFAULT '' NOT NULL,
  srcip varchar(16) DEFAULT '' NOT NULL,
  srcport varchar(6) DEFAULT '' NOT NULL,
  dstip varchar(16) DEFAULT '' NOT NULL,
  dstport varchar(6) DEFAULT '' NOT NULL,
  PRIMARY KEY (id)
);

$ cat mysql.schema | mysql gcia


cat insert_alert.pl
#!/usr/bin/perl -w

use DBI;
$driver = "mysql";
$dsn = "DBI:$driver:database=gcia;host=localhost;";

$dbh = DBI->connect($dsn, "root", "xksEIR187rllp");

@files=('alert');
foreach $file (@files) {
  #print $file . "\n";
  open ALERT, $file || die "Coudn't open $file";
  while ($line=<ALERT>){
    if ($line =~ /^(\d+)\/(\d+)-(\d+:\d+:\d+).\d+\s+\[\*\*\] spp_portscan: End of portscan from ([\d\w\W.]+):
TOTAL time\(\d+\w\) hosts\((\d+)\) TCP\((\d+)\) UDP\((\d+)\) [\w\s]*\[\*\*\]\s$/){
      #Portscan
      # $1 = Month, $2=date, $3=time, $4 = srcip, $5 = nohost, $6 = TCPno, $7 = UDPno
      $sql = "INSERT INTO portscan (timeof, srcip, nohost, tcpno, udpno) VALUES (" ;
      $sql .= "'2004-$1-$2 $3', '$4', '$5', '$6', '$7')";
      my $sth = $dbh->prepare( $sql ) or die "Couldn't prepare statement: " . $dbh->errstr;;
      $sth->execute();
      #print "$sql\n";
    } elsif ($line =~ /^(\d+)\/(\d+)-(\d+:\d+:\d+).\d+\s+\[\*\*\]\s([\s\W\w]+)\s\[\*\*\] ([\d.]+):(\d+) ->
([\d.]+):(\d+)\s$/) {
      #Rules alert
      # $1 = Month, $2=date, $3=time, $4 = Action, $5 = srcip, $6 = srcport, #7 = dstip, #8 = dstport
      $sql = "INSERT INTO alert (timeof, action, srcip, srcport, dstip, dstport) VALUES (" ;
      $sql .= "'2004-$1-$2 $3', 'REPLACEMEPLEASE', '$5', '$6', '$7', '$8')";
      #Hack to remove single quote from some action descriptions
      my $action = $4;
      $action =~ s/'/ /;
      $sql =~ s/REPLACEMEPLEASE/$action/;
      my $sth = $dbh->prepare( $sql ) or die "Couldn't prepare statement: " . $dbh->errstr;;
      $sth->execute();
      #print "$action\n";
      #print "$sql\n";
    # Match for alerts with no port numbers
    } elsif ($line =~ /^(\d+)\/(\d+)-(\d+:\d+:\d+).\d+\s+\[\*\*\]\s([\s\W\w]+)\s\[\*\*\] ([\d.]+) ->
([\d.]+)\s$/) {
      # $1 = Month, $2=date, $3=time, $4 = Action, $5 = srcip, $6 = dstip
      $sql = "INSERT INTO alert (timeof, action, srcip, srcport, dstip, dstport) VALUES (" ;
      $sql .= "'2004-$1-$2 $3', '$4', '$5', '99999', '$6', '99999')";
      my $sth = $dbh->prepare( $sql ) or die "Couldn't prepare statement: " . $dbh->errstr;;
      $sth->execute();
      #print "$sql\n";
    # Match for alerts with one port number
    } elsif ($line =~ /^(\d+)\/(\d+)-(\d+:\d+:\d+).\d+\s+\[\*\*\]\s([\s\W\w]+)\s\[\*\*\] ([\d.]+):(\d+) ->
([\d.]+)\s$/) {
      # $1 = Month, $2=date, $3=time, $4 = Action, $5 = srcip, $6 = srcport, #7 = dstip
      $sql = "INSERT INTO alert (timeof, action, srcip, srcport, dstip, dstport) VALUES (" ;
      $sql .= "'2004-$1-$2 $3', '$4', '$5', '$6', '$7', '99999')";
      my $sth = $dbh->prepare( $sql ) or die "Couldn't prepare statement: " . $dbh->errstr;;
      $sth->execute();
      #print "$sql\n";
    # Match for alerts with one IP
    } elsif ($line =~ /^(\d+)\/(\d+)-(\d+:\d+:\d+).\d+\s+\[\*\*\]\s([\s\W\w]+)\s\[\*\*\] ([\d.]+)\s$/) {
```

```
      # $1 = Month, $2=date, $3=time, $4 = Action, $5 = srcip
      $sql = "INSERT INTO alert (timeof, action, srcip, srcport, dstip, dstport) VALUES (" ;
      $sql .= "'2004-$1-$2 $3', '$4', '$5', '99999', '999.999.999.999', '99999')";
      my $sth = $dbh->prepare( $sql ) or die "Couldn't prepare statement: " . $dbh->errstr;;
      $sth->execute();
      #print "$sql\n";
    }else{
      #use for testing to print lines that do not match
      #print "$line";
    }
  }
}

./insert_alert.pl
```

With the alert data input into the SQL database it can now be analyzed. A combination of SQL queries and UNIX command line utilities were used to generate the data listed in the Network Statistics section. The commands run are listed here.

```
echo 'select action from alert order by action;' | mysql gcia | uniq -c | sort -n
echo 'select distinct srcip from alert order by srcip;' | mysql gcia | wc -l
echo 'select distinct srcip from alert where srcip LIKE "257.257%" order by srcip;' |
mysql gcia | wc -l
echo 'select distinct srcip from alert where srcip NOT LIKE "257.257%" order by srcip;' |
mysql gcia | wc -l
echo 'select srcip from alert where srcip LIKE "257.257.%" order by srcip;' | mysql gcia |
uniq -c | sort -nr | head -n10
echo 'select distinct srcip,action from alert where srcip = "EACH TOP 5 IP HERE" order by
srcip;' | mysql gcia | tail -n+2 |wc -l
echo 'select distinct srcip,action from alert where srcip = "EACH TOP 5 IP HERE" order by
srcip;' | mysql gcia
echo 'select srcip from alert where srcip NOT LIKE "257.257.%" order by srcip;' | mysql
gcia | uniq -c | sort -nr | head -n10
echo 'select distinct srcip,action from alert where srcip = "EACH TOP 5 IP HERE";' | mysql
gcia | tail -n+2 | wc -l
echo 'select distinct srcip,action from alert where srcip = "EACH TOP 5 IP HERE";' | mysql
gcia
echo 'select distinct dstip from alert order by dstip;' | mysql gcia | wc -l
echo 'select distinct dstip from alert where dstip LIKE "257.257.%" order by dstip;' |
mysql gcia | wc -l
echo 'select distinct dstip from alert where dstip NOT LIKE "257.257.%" order by dstip;' |
mysql gcia | wc -l
echo 'select dstip from alert where dstip LIKE "257.257.%" order by dstip;' | mysql gcia |
uniq -c | sort -nr | head -n10
echo 'select dstip from alert where dstip NOT LIKE "257.257.%" order by dstip;' | mysql
gcia | uniq -c | sort -nr | head -n10
echo 'select distinct dstport from alert order by dstport;' | mysql gcia | wc -l
echo 'select distinct dstport from alert where dstip LIKE "257.257.%" order by dstport;' |
mysql gcia | wc -l
echo 'select distinct dstport from alert where dstip NOT LIKE "257.257.%" order by
dstport;' | mysql gcia | wc -l
echo 'select dstport from alert where dstip LIKE "257.257.%" order by dstport;' | mysql
gcia | uniq -c | sort -nr | head -n10
echo 'select dstport from alert where dstip NOT LIKE "257.257.%" order by dstport;' |
mysql gcia | uniq -c | sort -nr | head -n10
echo 'select distinct srcport from alert order by srcport;' | mysql gcia | wc -l
echo 'select distinct srcport from alert where srcip LIKE "257.257.%" order by srcport;' |
mysql gcia | wc -l
echo 'select distinct srcport from alert where srcip NOT LIKE "257.257.%" order by
srcport;' | mysql gcia | wc -l
echo 'select srcport from alert where srcip LIKE "257.257.%" order by srcport;' | mysql
gcia | uniq -c | sort -nr | head -n10
echo 'select srcport from alert where srcip NOT LIKE "257.257.%" order by srcport;' |
mysql gcia | uniq -c | sort -nr | head -n10
echo 'select srcip from portscan where srcip LIKE "257.257.%" order by srcip;' | mysql
gcia | uniq -c | sort -nr | head -n5
```

```
echo 'select srcip from portscan where srcip NOT LIKE "257.257.%" order by srcip;' | mysql
gcia | uniq -c | sort -nr | head -n5
```

Scan Logs

The same basic procedure was then used on the scan logs. They were concatenated into a single file using a command loop. A MySQL database was created and a perl script was used to insert the records into the database.

```
for f in $(ls scans.*); do cat $f >> scans; done

ls -lh
total 955M
-rw-r--r--  1 user group 477M Feb 20 20:55 scans
-rw-r--r--  1 user group 89M Feb  9 14:41 scans.030915
-rw-r--r--  1 user group 75M Feb  9 14:41 scans.030917
-rw-r--r--  1 user group 172M Feb  9 14:42 scans.030918
-rw-r--r--  1 user group 143M Feb  9 14:42 scans.030919

cat mysql.schema
DROP TABLE IF EXISTS scan;
CREATE TABLE scan (
  id int(11) DEFAULT '0' NOT NULL auto_increment,
  timeof varchar(24) DEFAULT '' NOT NULL,
  srcip varchar(16) DEFAULT '' NOT NULL,
  srcport varchar(16) DEFAULT '' NOT NULL,
  dstip varchar(16) DEFAULT '' NOT NULL,
  dstport varchar(16) DEFAULT '' NOT NULL,
  prot varchar(16) DEFAULT '' NOT NULL,
  flags varchar(64) DEFAULT '' NOT NULL,
  PRIMARY KEY (id)
);

cat mysql.schema | mysql gcia

cat insert_scan.pl
#!/usr/bin/perl -w

use DBI;
$driver = "mysql";
$dsn = "DBI:$driver:database=gcia;host=localhost;";

$dbh = DBI->connect($dsn, "root", "PASSWORDHERE");

@files=('scans');
foreach $file (@files) {
  #print $file . "\n";
  open ALERT, $file || die "Coudn't open $file";
  while ($line=<ALERT>){
    #scan UDP
    if ($line =~ /^([\w]+) (\d+) (\d+:\d+:\d+) ([\d.]+):(\d+) -> ([\d.]+):(\d+) UDP\s*$/){
      # $1 = Month $2=date $3=time $4=srcip $5=srcport $6=dstip $7=dstport
      $sql = "INSERT INTO scan (timeof, srcip, srcport, dstip, dstport, prot) VALUES (" ;
      $sql .= "'2003-$1-$2 $3', '$4', '$5', '$6', '$7', 'UDP')";
      my $sth = $dbh->prepare( $sql ) or die "Couldn't prepare statement: " . $dbh->errstr;;
      $sth->execute();
      #print "$sql\n";
    #scan TCP
    }elsif ($line =~ /^([\w]+) (\d+) (\d+:\d+:\d+) ([\d.]+):(\d+) -> ([\d.]+):(\d+) ([\w]+
.+)\s*$/){
      # $1 = Month $2=date $3=time $4=srcip $5=srcport $6=dstip $7=dstport $8=flags
      $sql = "INSERT INTO scan (timeof, srcip, srcport, dstip, dstport, prot, flags) VALUES (" ;
      $sql .= "'2003-$1-$2 $3', '$4', '$5', '$6', '$7', 'TCP', '$8')";
      my $sth = $dbh->prepare( $sql ) or die "Couldn't prepare statement: " . $dbh->errstr;;
      $sth->execute();
      #print "$sql\n";
    }else{
      #use for testing to print lines that do not match
      #print "$line";
    }
```

```
  }
}

./insert_scan.pl
```

Numerous statistics were then generated using SQL queries and command line tools. Some simple DNS queries were run with the "host" command to verify some of the results. It appears that legitimate traffic may have been flagged as scans so an attempt was made to remove that traffic. Here are queries that revealed a DNS server.

```
echo 'select srcip from scan where srcip LIKE "130.85.%" order by srcip;' | mysql gcia |
uniq-c | sort -nr | head -n5
2597670 130.85.1.3
1521633 130.85.84.194
1315532 130.85.70.53
 580124 130.85.1.5
 356322 130.85.80.243

echo 'select dstport,prot from scan where srcip LIKE "130.85.1.3" order by prot,dstport;'
| mysql gcia | uniq -c | sort -nr | head -n5
2587353 53      UDP
   7289 123     UDP
    255 61005   UDP
    172 53      TCP
    133 61872   UDP

echo 'select dstip,dstport,prot from scan where srcip LIKE "130.85.1.3" order by
dstip,dstport,prot;' | mysql gcia | uniq -c | sort -nr | head -n5
  55787 192.26.92.30    53      UDP
  46178 203.20.52.5     53      UDP
  44379 130.94.6.10     53      UDP
  39550 192.52.178.30   53      UDP
  35848 205.231.29.244  53      UDP

host 192.26.92.30
30.92.26.192.in-addr.arpa domain name pointer c.gtld-servers.net.
host 203.20.52.5
Host 5.52.20.203.in-addr.arpa not found: 3(NXDOMAIN)
host 130.94.6.10
10.6.94.130.in-addr.arpa domain name pointer bsp1.bondedsender.org.
host 192.52.178.30
30.178.52.192.in-addr.arpa domain name pointer k.gtld-servers.net.
host 205.231.29.244
244.29.231.205.in-addr.arpa domain name pointer a.ns.dsbl.org.
```

Here are queries that uncover what appears to be NetBIOS traffic between internal hosts. Note that unlike the alert logs the scan logs have not obfuscated the IP addresses.

```
echo 'select dstport,prot from scan where srcip LIKE "130.85.70.53" order by
dstport,prot;' |mysql gcia | uniq -c | sort -nr | head -n5
1225687 135     TCP
  89844 80      TCP
      1 dstport prot
      1 443     TCP

echo 'select dstip,dstport,prot from scan where srcip LIKE "130.85.70.53" order by
dstip,dstport,prot;' | mysql gcia | uniq -c | sort -nr | head -n5
     40 130.84.173.131  135     TCP
     38 130.84.111.115  135     TCP
     38 130.84.110.173  135     TCP
```

```
     37 130.84.44.120   135     TCP
     37 130.84.228.44   135     TCP
```

A command loop was then run to return the desired statistical information. The first run detected DNS, SMTP, and NetBIOS traffic. A subsequent loop was run removing these from the results. The filtered loop is listed here because it is more complex and based on the unfiltered loop. A similar look was then run with "WHERE srcip NOT LIKE '130.85.%'" to retrieve the same statistics for external sources.

```
for i in $(echo 'select srcip from scan where srcip LIKE "130.85.%" and dstport != "53"
and dstport != "135" and dstport != "25" and dstport != "137" order by srcip;' | mysql
gcia | uniq -c | sort -nr | head -n5 | cut -c9-100); do
echo
echo $i
echo "select dstport,prot from scan where srcip LIKE '$i' order by dstport,prot;" | mysql
gcia | uniq -c | sort -nr | head -n5
echo "select dstip,dstport,prot from scan where srcip LIKE '$i' order by
dstip,dstport,prot;" | mysql gcia | uniq -c | sort -nr | head -n5; done

130.85.70.53
1225687 135     TCP
  89844 80      TCP
      1 dstport prot
      1 443     TCP
     40 130.84.173.131  135     TCP
     38 130.84.111.115  135     TCP
     38 130.84.110.173  135     TCP
     37 130.84.44.120   135     TCP
     37 130.84.228.44   135     TCP

130.85.70.207
   2401 33774   UDP
   2056 65090   UDP
   1997 18304   UDP
   1978 11564   UDP
   1808 33513   UDP
   2256 24.224.188.141  33774   UDP
   2056 12.222.24.242   65090   UDP
   1997 69.27.66.247    18304   UDP
   1978 142.217.110.70  11564   UDP
   1808 81.152.67.233   33513   UDP

130.85.84.210
  26394 6257    UDP
    426 80      TCP
    149 16257   UDP
     83 6256    UDP
     77 26257   UDP
     98 172.133.50.228  6257    UDP
     90 62.211.252.6    6257    UDP
     83 68.72.171.140   6257    UDP
     78 68.14.89.26     6257    UDP
     69 80.116.225.5    6257    UDP

130.85.98.132
  12101 22321   UDP
   4289 7674    UDP
     14 80      TCP
      5 1025    UDP
      4 2865    UDP
      9 220.122.198.101 22321   UDP
      8 211.212.96.4    22321   UDP
```

```
      7 220.91.73.39    22321   UDP
      7 220.81.215.195  22321   UDP
      7 219.248.155.236 22321   UDP

130.85.82.2
   1626 4046    UDP
   1621 32801   UDP
   1605 2645    UDP
   1594 1173    UDP
   1553 1100    UDP
   1626 68.34.149.132   4046    UDP
   1621 4.3.172.73      32801   UDP
   1605 24.209.234.234  2645    UDP
   1594 24.197.242.66   1173    UDP
   1553 24.79.219.29    1100    UDP
```

Out of Specification Logs

The out of specification (OOS) logs were handled in a different way. First they were concatenated like the others but then there were analyzed in their native ASCII format. A perl script was used to generate some basic statistics on the data. The file is a list of packets with their IP and TCP headers decoded. Any data in the packet is then listed in HEX and ASCII columns. The perl script gives the number of packets based on the datagram lengths. Any packets larger 64 bytes are printed to screen. After viewing the results a filter was added to match the keyword “Kazaa” in the packet. Here is the script and its statistical output.

```
for f in $(ls OOS*); do cat $f >> oos; done

grep '^09/..-..:..:' oos | wc -l
15356

wc -l oos
130595 oos

cat parse.pl

#!/usr/bin/perl

$/ = "=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+";
my $file = shift @ARGV;
my $dgm40 = '0';
my $dgm44 = '0';
my $dgm48 = '0';
my $dgm52 = '0';
my $dgm56 = '0';
my $dgm60 = '0';
my $dgm64 = '0';
my $kazaa = '0';
my $print = '0';

open F, "$file" or die("Cannot open $file $!");
while (<F>) {
        if (/DgmLen:40/) { $dgm40++; next; }
        if (/DgmLen:44/) { $dgm44++; next; }
        if (/DgmLen:48/) { $dgm48++; next; }
        if (/DgmLen:52/) { $dgm52++; next; }
        if (/DgmLen:56/) { $dgm56++; next; }
        if (/DgmLen:60/) { $dgm60++; next; }
        if (/DgmLen:64/) { $dgm64++; next; }
        if (/Kazaa/) { $kazaa++; next; }
        #if (/^.*\n.*\n.*\n.*\n.*\n.*\n.*\n.*$/) { print $_; next; }
        $print++;
        print $_;
}
print "\n40: " . $dgm40;
print "\n44: " . $dgm44;
```

```
print "\n48: " . $dgm48;
print "\n52: " . $dgm52;
print "\n56: " . $dgm56;
print "\n60: " . $dgm60;
print "\n64: " . $dgm64;
print "\nkazaa: " . $kazaa;
print "\nprint: " . $print;
print "\n";
close F;

chmod 700 parse.pl

./parse.pl oos |tail –n9
40: 341
44: 381
48: 136
52: 63
56: 33
60: 13589
64: 177
kazaa: 623
print: 10
```

List of References

James. Ntpdx overflow attempt sig triggered by ntpdc query. 14 December 2002
http://archives.neohapsis.com/archives/snort/2002-12/0413.html

Frasunek, Przemyslaw. ntpd =< 4.0.99k remote buffer overflow. 4 April 2001
http://www.securityfocus.com/archive/1/174011

The NetBSD Foundation, Inc. 2001-004: NTP remote buffer overflow. 5 April 2001
http://www.securityfocus.com/advisories/3200

Turkia, Miika. Level Two Intrusion Detection In Depth. 28 January 2001
http://www.giac.org/certified_professionals/practicals/gcia/0352.php

Stearns, William Ramen Worm Detect Script. 15 February 2001
http://www.sans.org/y2k/ramen.htm

Internet Security Systems (ISS) X-Force. ISS-028: Windows Backdoor Update III. 6 July 1999
http://www.securityfocus.com/advisories/1644

Rao, Subba. Ramen worm and Snort log entry. 17 June 2001
http://www.securityfocus.com/archive/96/191589

Antirez. Dumb Scan. 18 December 1998
http://www.kyuzz.org/antirez/papers/dumbscan.html

LURHQ Threat Intelligence Group. Intrusion Detection: In-Depth Analysis.
http://www.lurhq.com/idsindepth.html

Rautiainen, Sami. F-Secure Virus Descriptions : Adore. April 2001
http://www.f-secure.com/v-descs/adore.shtml

Microsoft Corporation. INF: TCP Ports Needed for Communication to SQL Server Through a Firewall. 16 September 2003
http://support.microsoft.com/kb/q287932/

Microsoft Corporation. Using the MMS protocol. 2003
http://www.microsoft.com/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/Default.asp?url=/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/wmserver/wmsmmsservercontrolprotocol.asp

Microsoft Corporation. Using Windows 2000 with Service Pack 4 in a Managed Environment.
http://www.microsoft.com/technet/prodtechnol/Windows2000Pro/maintain/w2kmngd/14_2kwmp.mspx

Whitle. McAfee Guardian and Yahoo Launchcast. 17 April 2003
http://forums.mcafeehelp.com/viewtopic.php?t=8978

Flynn, Hal. FOCUS on Sun: Hardening Solaris - Creating a Diamond in the Rough. 2 October 2000
http://www.securityfocus.com/infocus/1365

Lam, Jason. GCIA Practical. 14 October 2001

http://www.giac.org/certified_professionals/practicals/gcia/0441.php