Global Information Assurance Certification Paper

SANS Practical
Network Security 2000
Intrusion Detection

Submitted by S.I. SHAW for GCIA

Assignment 1 Network Detects

Event Trace Correlations # 1

1. Source of Traces
Source of the traces were an internal web server

2. Detect Generated By
The detects were generated by Realsecure 3.X, which I chose as I am not familiar with the logging or alert format

3. Probability that Source Address was Spoofed

Low probability that these IP's are spoofed due to the fact that there are several correlations of malformed packets and Demon net having misconfigured software.

4. Attack Description

A standard IP packet contains an 8 bit protocol field. Common values for this field include 6 for tcp, 17 for udp and 1 for icmp. Attackers sometimes use a non standard value for this field in order to exchange data between machines without logging mechanisms detecting the data that is being transmitted. However RealSecure detected this activity and dumped the following alert:

8/11/99 6:31:00 AM	IPProtocol Violation	1157	HTTP	xxx.xxx.220.23	my.net.machine.3
8/11/99 6:31:00 AM	IPProtocol Violation	1157	HTTP	xxx.xxx.220.23	my.net.machine.3
8/11/99 6:31:21 AM	IPProtocol Violation	17465	HTTP	xxx.xxx.220.23	my.net.machine.3
8/11/99 6:31:21 AM	IPProtocol Violation	17465	HTTP	xxx.xxx.220.23	my.net.machine.3
8/11/99 7:59:14 AM	IPProtocol Violation	41016	HTTP	xxx.xxx.220.23	my.net.machine.3
8/11/99 7:59:14 AM	IPProtocol Violation	41016	HTTP	xxx.xxx.220.23	my.net.machine.3

5. Attack Mechanism
Not really an attack, however I wanted to present this one because, originally, I had no idea what this traffic was and spent some time trying to figure it out. While I do not recommend basing all your analysis on correlations, I do believe in this instance that due to the lack of action on behalf of Demon net that all an intrusion detection analyst can do is use what other people have found. The automatic, generated responses, that people are receiving from Demon net is proof that this type of malformed traffic will probably continue for some time.

6. Correlations
Several correlations to this activity

www.sans.org/y2k/practical/Markus_DeShon.html
Wherein it is stated that "Some normally suspicious activity was detected from www.demon.co.uk, including a number of badly formed packets. However demon.co.uk is known to be the source of odd network activity due to a bad router --this activity is most likely innocuous." There is a standard reply that is sent from Demon that suggests that they have taken this issue up with their supplier.

7. Evidence of Active Targeting

No evidence of active targeting more likely associated with false positive.

8. Severity

Formula to calculate the Severity = (Target Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures) This formula is not likely to add anything to the analysis due to the fact that this is a false positive and the packet filtering device throws away the IPProtocol Violation traffic.

9. Defensive Recommendations
The only defensive recommendation that I can offer is to continue to drop this type of traffic and it does not hurt to continue to send futile messages to Demon net with specific examples of their malformed traffic.

10. Test Question

What is the IP protocol for UDP TCP and ICMP
a) 10, 6 ,4
b) 1, 6 ,7
c)6, 7, 1
d)6, 17, 1

answer is d

Event Trace Correlations # 2

1. Source of Traces

Protected customer network, which was running a newer version of IMAIL and therefore was not vulnerable to this version of the IMAIL buffer overflow

2. Detect Generated By

Net Ranger sensor 2.X

3. Probability that Source Address was Spoofed
Low probability that the source address was spoofed because it was a coordinated hit not a scan for services, therefore I am assuming that the attacker was looking to exchange some data.

4. Attack Description

2000/04/10 11:21:10 OUT IN 3526 IMAP xxx.xxx.xxx.7 xxx.xxx.1.98 41554 00143 IMAP

IP's were obfuscated to protect the innocent and the guilty. Time / Date -> Direction -> Signature Number and Name ->Attacker -> Victim -> Originating Port -> Destination Port -> Identified Port if known

3526 IMAP Buffer Overflow: This signature triggers on receipt of packets bound for port 143 that are indicative of an attempt to overflow the IMAP daemon user buffer. This may be the precursor to an attempt to gain unauthorized access to the system resources.

5. Attack Mechanism

There are several IMAP overflows and since we can not look at the signature that pulled this packet from the line and analyse the contents I can only discuss some of the more common overflow exploits. I thought I might introduce an older exploit, the nessus plugin that allows you to gain root. The script was written by Renaud Deraison and is based on the cve 1999 0005 for Imail' s imap buffer overflow.

6. Correlations
No correlations of the above mentioned attacker, however there are several correlations to other IP's targeting IMAP with Buffer Overflows. Having said that the attacker only showed up once in the logs with no other apparent attempts to penetrate our systems and

therefore correlations with other IMAP overflow attempts would not offer anything to this analysis. For further information on IMAP overflows check out. www.mitre.org and their cve database, where you will find well known IMAP vulnerability entries in CVE-1999-0005, CVE-1999-0042, CVE-1999-0920 and CVE-2000-0053. Worth mentioning is that IMAP is listed as #9 in the "Ten Most Critical Internet Security Threats" from the www.sans.org/topten.htm .

7. Evidence of Active Targeting
Given the fact this packet was pulled because of an apparent attempt to overflow the buffer and possibly can root access, it could be said that we were actively targeted, at least in the sense that there are no other correlations with other servers in the network and given the fact that we have not seen any other activity from this host. There was no evidence in the logs of fingerprinting or scanning which is not surprising given the fact that the attacker was not using the correct exploit. Interestingly enough the 1.98 host resolves to a chat server associated with a University.

8. Severity

Formula to calculate the Severity = (Target Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures)

(2 + 1) – (5 + 5) = -2

9. Defensive Recommendations
All patches and updates were in place, however I still recommend dumping these types of packets every once in a while to make sure that there is not a new vulnerability but I just never get the chance. It is also good measure to be watching bugtraq and other lists of the kind if you want to keep up with the latest exploits and vulnerabilities. Given that this site is an ecommerce site it maybe a good idea to prudently watch the mail servers as it contains information of a critical business nature, more so, say then a workstation were the internal classifieds are posted.

10. Test Question

The IMAP port is

a) 110
b) 43
c) 143
d) 113

the answer is c

Event Trace Correlations # 3

1. Source of Traces

Ecommerce client network

2. Detect Generated By

Real Secure

3. Probability that Source Address was Spoofed
Low probability that the source address was spoofed as the attacker would need to receive a response back from the victim. Also given the fact that the signature fires after the receipt of overflow packets, I assume that there was a handshake

4. Attack Description

2000/04/10 11:21:10 OUT IN 3550 POP_Overflow xxx.xxx.xxx.23 xxx.xxx.123.101 01334 00110 POP3

IP's were obfuscated to protect the innocent and the guilty. Tine / Date -> Direction -> Signature Number and Name ->Attacker -> Victim -> Originating Port -> Destination Port -> Identified Port if known

3550 POP Buffer Overflow: This signature triggers on receipt of packets bound for port 110 that are indicative of an attempt to

overflow the POP daemon user buffer.
This may be the precursor to an attempt to gain unauthorized access to the system resources.

5. Attack Mechanism
There are several POP3 overflow programs to be found at packetstorm.securify.com/9912-exploits. Rewted Network Security Labs found a local/remote DoS attack in PakMail and POP3 servers. The buffer overflow is caused by a long username specified for the RCPT TO field in the SMTP server. The buffer length is approximately 1390 characters and will crash with the following error message. PAKMAIL caused an invalid page fault in module KERNEL32.DLL at 0137:bff9a5d0. Similarly the POP3 server is vulnerable to the same attack except that the buffer overflow is caused when an extra long 'pass' field is entered. The buffer is approximately 1400 characters

6. Correlations
This scan, as the above IMAP overflow attempt, is also found in the "Ten Most Critical Internet Security Threats" from the www.sans.org/topten.htm I also found an excellent resource on the web for articles on mail overflows which can be found at ntsecurity.win2000mag-asap.com. There were no correlations for the above mentioned attacker IP, however this exploit is so common that I am sure that there could be several correlations, but many people get so tired of seeing mail overflow attempts on their systems even when they have the newer version of mail that has no published vulnerability…yet.

7. Evidence of Active Targeting
There was evidence of targeting, but not very good targeting as the overflow attack was intended for an older version of the vulnerability.

8. Severity

Formula to calculate the Severity = (Target Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures)
(2 + 1) – (5 + 5) = -2
The buffer overflow was directed at a system that was already protected with the appropriate patches

9. Defensive Recommendations

Latest version of POP3 server is already installed, however given the fact that this is an ecommerce client recommendation to flag the IP for future trend analysis is recommended. Christmas is just around the corner and I would rather be proactive in this instance as the mail server is a prime target for gathering client base information. My analysis is that this is a hacker with little knowledge of fingerprinting and discovery, however if I see him / her come back with an exploit that fits the target service I would want to know.

10. Test Question

The port for POP3 is
a) 143
b) 111
c) 110
d) 5
the answer is c

Event Trace Correlations # 4

1. Source of Traces
Standard install of workstations behind a firewall and IDS sensor

2. Detect Generated By
Net Ranger

3. Probability that Source Address was Spoofed
In order for the buffer overflow data to get to the victim there needs to be a 3 way handshake

4. Attack Description

2000/04/04 10:14:17 IN OUT 3650 SSH RSAREF2 Buffer O 207.107.11.38 149.99.142.28 01067 000222 ssh

Netranger dumped this alert based on the RSAREF2 buffer overflow first released by core sdi (www.core-sdi.com/advisories/buffer%20overflow%20ing.htm.

5. Attack Mechanism

Systems running some versions of sshd as well as some SSL enabled web servers using the RSAREF2 product (via the --with-rsaref option) are vulnerable to buffer overflows. As described in the CERT Advisory CA 1999 15 (www.cert.org/advisories/CA-1999-15.html) , utilisation of the two vulnerabilities mentioned above allows an intruder to execute arbitrary code with privileges of the process running sshd, typically root.

6. Correlations

No correlations for the attacker IP and no previous correlations with our network logs.

7. Evidence of Active Targeting
The attacker did not show up in our logs before the attempted intrusion and he did target only one machine in particular, however we are not running ssh on that machine. No evidence of active targeting, it might have been an anomaly

8. Severity

Formula to calculate the Severity = (Target Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures)

(5 + 3) – (4 + 4) = 0

Since we are not running ssh and the countermeasures picked off the attempt we can be fairly confident in our security posture. However we did take a look to see if the attacker had attempted to do this or any other type of attacks against our system just to make sure that we were not missing anything in relation to our malicious visitor.

9. Defensive Recommendations

This system was running fully patched, however for other systems that may not have been patched you should know that in order for the exploit to work the vulnerability requires an application program to call the rsaref2 library and input malicious data. As suggested in the CERT Advisory it may be possible to prevent this through input validation. For further patch information try RSA Security.

10. Test Question
SSL and SSH mean

a) Slow Secure Link and Secure Shell
b) System Support Layer and System Support Host
c) Secure Socket Layer and Secure Shell
d) System Support Layer and Security Services Host

The answer is c

Event Trace Correlations # 5

1. Source of Traces
Cable modem traffic

2. Detect Generated By

Network Ice (www.networkice.com)

3. Probability that Source Address was Spoofed

Good probability that source was not spoofed, however if this was a concern we could take a look to see if there were any reset packets and make a fair deduction on who is the real attacker, however in this case with the data that we have obtained from the network ice sensor this is not a suitable path to follow.

4. Attack Description

2000-09-27 11:23:50	TCP OS Fingerprint	200.36.46.3	port=111&flags=SF&options	resolves to the Network Information Center of Mexico
2000-10-25 01:37:33	TCP OS Fingerprint	211.36.35.125	port=9704&flags=SF&options	resolves to Korean Network Information Center
2000-10-25 01:37:33	TCP OS Fingerprint	24.27.28.173	port=9704&flags=SF&options	resolves to cs2728-173.Austin.rr.com
2000-10-27 18:19:15	TCP OS Fingerprint	62.98.51.57	port=111&flags=SF&options	resolves to Wind Telecommunications in Italy
2000-11-07 12:55:20	TCP OS Fingerprint	200.36.46.3	port=111&flags=SF&options	resolves to the Network Information Center of Mexico
2000-11-09 01:47:51	TCP OS Fingerprint	64.122.30.11	port=111&flags=SF&options	resolves to Integra Telecom of Oregon
2000-11-09 09:14:42	TCP OS Fingerprint	211.36.35.125	port=111&flags=SF&options	resolves to Korean Network Information Center

A good resource for information on OS Fingerprinting is Lance Spitzner 's article on Passive Fingerprinting found at http://atls.spaceports.com/~secure66/finger.htm. Spitzner suggests that there are four basic signatures that can be used to identify an operating system. According to the paper cited above ttl, window size, the do not fragment bit and the type of service, one can identify an operating system with reasonable success. One of many signature databases for identifying operating systems that has popped up on the internet can be found at www.enteract.com

5. Attack Mechanism

Many different attack mechanisms can be used for example, winfingerprint, nmap and others. For discussion purposes we can leave the type of tool to the exterior and focus on what type of information the attackers are trying to uncover. Information gleaned from www.insecure.org/nmap/nmap-fingerprinting-article.html describes the act of querying the TCP/IP stack and what information can be retrieved. As an example of the power of nmap OS fingerprinting and the information that it gleans an nmap -sS -F -o -O www.victim/24 would syn scan on known ports from the etc/services, log the results to victim.log, be verbose, do an OS scan and scan the Class C of victim.com resulting in a pretty accurate detection of OS's

6. Correlations
There are several correlations for the above mentioned IP's using OS fingerprinting against other hosts.

Correlations for 200.36.46.3
Global Incident Analysis Center: Detects Analyzed 9/5/00
... - [root@pilsner 200.36.46.3]# cat TCP:111-111 [**] SCAN-SYN FIN [**] 09/02-10:03:47.923804 200.36.46.3:111 -> 136.142.91.xxx:111 TCP TTL:29 TOS:0x0 ID:39426 ...
www.sans.org/y2k/090500.htm - 17k

harshbutfair.org: portscans

... Sep 12 04:08, tcp, 200.36.46.3, 111 portmap, SANBORNS SA de CV, Mexico. Sep 12 03:29, udp, 149.142.196.164, 137 netbios-ns, UCLA Campus Network Services, USA. ...
www.harshbutfair.org/portscan.html - 19k

Neohapsis Archives - Incidents list - sunrpc - From rgg@ ...
... two of my servers: Aug 15 22:00:41 desktop abacus_sentry[676]: attackalert: FIN stealth scan from host: tlacaelel.sanborns.com.mx/200.36.46.3 to TCP port: 111. ...
archives.neohapsis.com/archives/incidents/2000-08/0148.html - 5k

Security Incidents Reporting mailing list archive sunrpc
... two of my servers: Aug 15 22:00:41 desktop abacus_sentry[676]: attackalert: FIN stealth scan from host: tlacaelel.sanborns.com.mx/200.36.46.3 to TCP port: 111. ...
lists.insecure.org/incidents/2000/Aug/0183.html - 10k –

Interrorem: Mailing List Archive
... two of my servers: Aug 15 22:00:41 desktop abacus_sentry[676]: attackalert: FIN stealth scan from host: tlacaelel.sanborns.com.mx/200.36.46.3 to TCP port: 111. ...
www.interrorem.com/arch/incidents/08/0187.php3 - 19k

Correlations for 24.27.28.173

http://www.sans.org/y2k/102800.htm

Oct 25 11:40:58 24.27.28.173:9704 -> 192.168.30.1:9704 SYNFIN **SF****

asctcpdump replay

11:40:58.771141 24.27.28.173.9704 > 192.168.30.1.9704: SF

875218391:875218391(0) win 1028 (ttl 22, id 39426)

4500 0028 9a02 0000 1606 38a3 181b 1cad E..(.....8.....

xxxx xxxx 25e8 25e8 342a c5d7 45c0 56dc .p..%.%.4*..E.V.

5003 0404 f743 0000 0000 0000 0000 P....C.....

http://www.sans.org/y2k/102600.htm

Oct 24 17:40:49 cc1014244-a kernel: securityalert: tcp if=ef0 from 24.27.28.173:9704 to 24.3.21.199 on unserved port 9704

An attempt to exploit 9704 and spawn a shell is also discussed at www.sans.org/y2k/082200.htm

7. Evidence of Active Targeting

There can be a case made that just the act of OS fingerprinting is active targeting in amongst itself, be it that attacker used a random generated list of victim IP's or not. In this instance of OS fingerprinting there is even more cause for concern given the fact that xxx.xxx.46.33 has come back for another round of OS fingerprinting, two months later. Either that or we all should be looking for this guy because his IP list is so large that it takes two months for it to cycle through.

8. Severity

Formula to calculate the Severity = (Target Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures)

(5 + 4) – (2 + 2) = 5

I only have one computer and the attacks, as presented in the correlation data could be part of a broader attempt to scan several hosts looking for vulnerabilities. A recheck of log data may show that this is not the only vulnerability that is coming out of Mexico. This IP has been added to the watchlist for future activity reports.

9. Defensive Recommendations

OS fingerprinting is a very stealthy, but you can limit the results by limiting the type of information that you send back. There are ways that you can fool NMAP's OS fingerprinting, which is commonly referred to as port forwarding. One example of this can be found at www.legionsunderground.org/helo1.txt where ports 21, 80, 113 and 139 on a linux firewall were forwarded to an OpenBSD box.

10. Test Question

OS Fingerprinting is:

a) a way in which OS developers are working with law enforcement to identify hackers
b) only theoretical at this point
c) can only be executed on windows based machines
d) a common way to avoid some IDS sensors

the answer is d

Section 2 Analysis of an Attack

Distributed Denial of Service attack tool TFN2k
- Downloaded from packetstorm.securify.com
- Tested on engineering lab test bed containing 192.168.1.1, 192.168.1.2, 192.168.1.10, 192.168.1.20 and 192.168.1.30
- Attacker 192.168.1.30
- Agents 192.168.1.10 and 192.168.1.40
- Victim 192.168.1.2
- Tcpdump listening in passive mode dumping to file 192.168.1.1

First communication on the network is an ARP request: who has .10

16:45:41.135155 arp who-has 192.168.1.10 tell 192.168.1.30
16:45:41.135274 arp reply 192.168.1.10 is-at 0:50:da:ca:44:f0

11 packets of commands issued to .10 which varies depending on the speed of the computers and how many attackers are utilized. It should be noted that you would not likely see this type of communication as the agents would be distributed over the internet and your promiscuous mode sniffer would not see this. The attacker or master is communicating to several agents or slaves in this case to 1.10 and 1.40 and issuing the command to do an icmp flood. Also not that even in the test bed the master that issues the command has it's IP spoofed as well.

16:45:41.135419 151.110.193.0.3173 > 192.168.1.10.53840: . 8744175:8744235(60) ack 0 win 9344
16:45:41.150847 151.110.193.0.33935 > 192.168.1.10.48576: S 0:60(60) ack 1177537 win 0
16:45:41.170639 151.110.193.0.48010 > 192.168.1.10.21060: udp 43
16:45:41.190711 151.110.193.0 > 192.168.1.10: icmp: echo reply
16:45:41.210811 151.110.193.0.58502 > 192.168.1.10.50022: S 0:60(60) ack 4263865 win 0
16:45:41.230841 151.110.193.0.10229 > 192.168.1.10.21633: S 0:60(60) win 0
16:45:41.250639 151.110.193.0.43994 > 192.168.1.10.60418: udp 43
16:45:41.270814 151.110.193.0.11421 > 192.168.1.10.54034: S 0:60(60) ack 2113512 win 0
16:45:41.290713 151.110.193.0 > 192.168.1.10: icmp: echo reply
16:45:41.310921 151.110.193.0.15370 > 192.168.1.10.37050: . 5223945:5224005(60) ack 6608064 win 56050
16:45:41.331048 151.110.193.0.2898 > 192.168.1.10.42104: S 0:60(60) ack 10300038 win 0

Who has the victim .2
16:45:41.345935 arp who-has 192.168.1.2 tell 192.168.1.10

Oh I have the victim and the hardware address is
16:45:41.346060 arp reply 192.168.1.2 is-at 0:50:da:ca:55:87

Thank you pizza delivery
16:45:41.346193 192.27.122.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.346298 1.122.60.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.346403 94.14.231.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.346508 123.166.39.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.346613 61.150.218.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.346718 19.163.141.0 > 192.168.1.2: icmp: echo request [ttl 0]

16:45:41.346823 156.217.57.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.346929 38.246.102.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.347034 7.191.42.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.347140 2.182.196.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.347245 92.125.207.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.347350 213.58.100.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.347455 104.95.122.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.347560 184.236.182.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.347665 134.234.17.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.347771 156.217.153.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.347876 196.199.95.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.347981 89.85.41.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.348086 241.124.129.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.348191 200.255.204.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.348296 49.244.55.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.348402 235.37.173.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.348507 217.16.241.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.348612 51.165.49.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.348717 37.187.11.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.348822 151.85.26.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.348928 17.220.54.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.349033 239.88.200.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.349138 11.184.56.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.349244 191.99.209.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.349350 60.175.63.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.349454 124.191.83.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.349560 40.101.167.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.349665 46.23.50.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.349773 115.147.119.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.349876 241.118.230.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.349981 61.73.252.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.350085 158.209.56.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.350190 50.125.153.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.350295 174.0.59.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.350401 166.202.9.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.350506 61.233.209.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.350611 38.218.199.0 > 192.168.1.2: icmp: echo request [ttl 0]

and the beat goes on

16:45:41.371230 167.149.51.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.371336 40.32.0.1 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.371441 4.196.79.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.371546 32.155.32.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.371651 96.21.125.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.371756 158.8.208.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.371862 174.19.151.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.371967 173.20.240.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.372072 25.162.95.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.372177 201.20.190.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:41.372282 112.70.125.0 > 192.168.1.2: icmp: echo request [ttl 0]

16:45:51.189545 159.243.190.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.189652 189.224.15.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.189755 108.203.190.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.189860 53.132.157.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.189965 200.111.86.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.190071 31.137.179.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.190176 166.189.108.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.190281 151.223.226.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.190386 10.142.97.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.190491 131.216.71.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.190597 39.215.34.0 > 192.168.1.2: icmp: echo request [ttl 0]

16:45:51.190702 102.68.115.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.190807 243.90.173.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.190912 89.185.108.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.191018 146.162.232.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.191123 186.187.205.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.191228 156.152.224.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.191333 187.183.154.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.191438 235.0.66.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.191543 71.34.232.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.191649 168.23.15.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.191754 160.111.239.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.191859 32.154.213.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.191964 58.23.57.0 > 192.168.1.2: icmp: echo request [ttl 0]
16:45:51.192070 252.209.77.0 > 192.168.1.2: icmp: echo request [ttl 0]

Command to cease flooding (-c0) issue from spoof address to the .10 machine
For analysis purposes it is not necessary to show the packet contents of the commands as they are encrypted. To further complicate packet analysis and detection decoy packets are also used. Encryption is Base 64 encoded and Cast 256 encrypted, which by the way was only excluded from the AES competition because of its speed not it's cipher strength. As a note, a paper presented by Jon Barlow and Woody Thrower March 7 (www.packetstorm.securify.com) suggests that there is a fingerprint that has been left behind by the encryption. As stated in the paper "We suspect it was the intent of the author to create variability in the length of each packet by padding each packet with one to sixteen zeroes. Base 64 encoding of the data translates this sequence of trailing zeroes into a sequence of 0x41's ('A'). There will always be at least one trailing 'A' and up to 16. This point was also broached during the tcpdump filter writing session.

16:45:51.197730 45.154.181.0.48883 > 192.168.1.10.27551: udp 43
16:45:51.197825 45.154.181.0.10095 > 192.168.1.10.52632: S 0:60(60) ack 0 win 62387
16:45:51.197910 45.154.181.0 > 192.168.1.10: icmp: echo reply

The trace below shows what happens when one of the agents does not respond or when you have inputted the wrong password. 1.40 slave did not flood the victim. You will not that there are twenty command packets issued to 1.40 to tell it to stop, which is written into the tfn2k code.

16:45:51.197996 175.66.65.0.64920 > 192.168.1.40.26982: udp 43
16:45:51.198090 175.66.65.0.57044 > 192.168.1.40.51775: S 5727161:5727221(60) win 58167
16:45:51.203390 175.66.65.0 > 192.168.1.40: icmp: echo reply
16:45:51.221167 175.66.65.0.14973 > 192.168.1.40.17125: S 16492938:16492998(60) ack 0 win 0
16:45:51.241182 175.66.65.0 > 192.168.1.40: icmp: echo reply
16:45:51.260966 175.66.65.0.3083 > 192.168.1.40.11981: udp 43
16:45:51.281283 175.66.65.0.55175 > 192.168.1.40.28100: S 3792565:3792625(60) ack 838716 win 2115
16:45:51.300957 175.66.65.0.46860 > 192.168.1.40.51568: udp 43
16:45:51.321093 175.66.65.0.34258 > 192.168.1.40.26803: S 0:60(60) ack 0 win 0
16:45:51.340966 175.66.65.0.19712 > 192.168.1.40.42493: udp 43
16:45:51.361110 175.66.65.0.5961 > 192.168.1.40.32750: udp 43
16:45:51.381056 175.66.65.0.49164 > 192.168.1.40.18273: udp 43
16:45:51.400994 175.66.65.0 > 192.168.1.40: icmp: echo reply
16:45:51.420986 175.66.65.0 > 192.168.1.40: icmp: echo reply
16:45:51.441193 175.66.65.0.34477 > 192.168.1.40.10262: S 0:60(60) win 37865
16:45:51.461020 175.66.65.0 > 192.168.1.40: icmp: echo reply
16:45:51.481024 175.66.65.0 > 192.168.1.40: icmp: echo reply
16:45:51.500959 175.66.65.0.15204 > 192.168.1.40.52062: udp 43
16:45:51.521105 175.66.65.0.40373 > 192.168.1.40.58892: . 0:60(60) ack 0 win 0
16:45:51.541099 175.66.65.0.59113 > 192.168.1.40.3617: S 0:60(60) ack 0 win 0

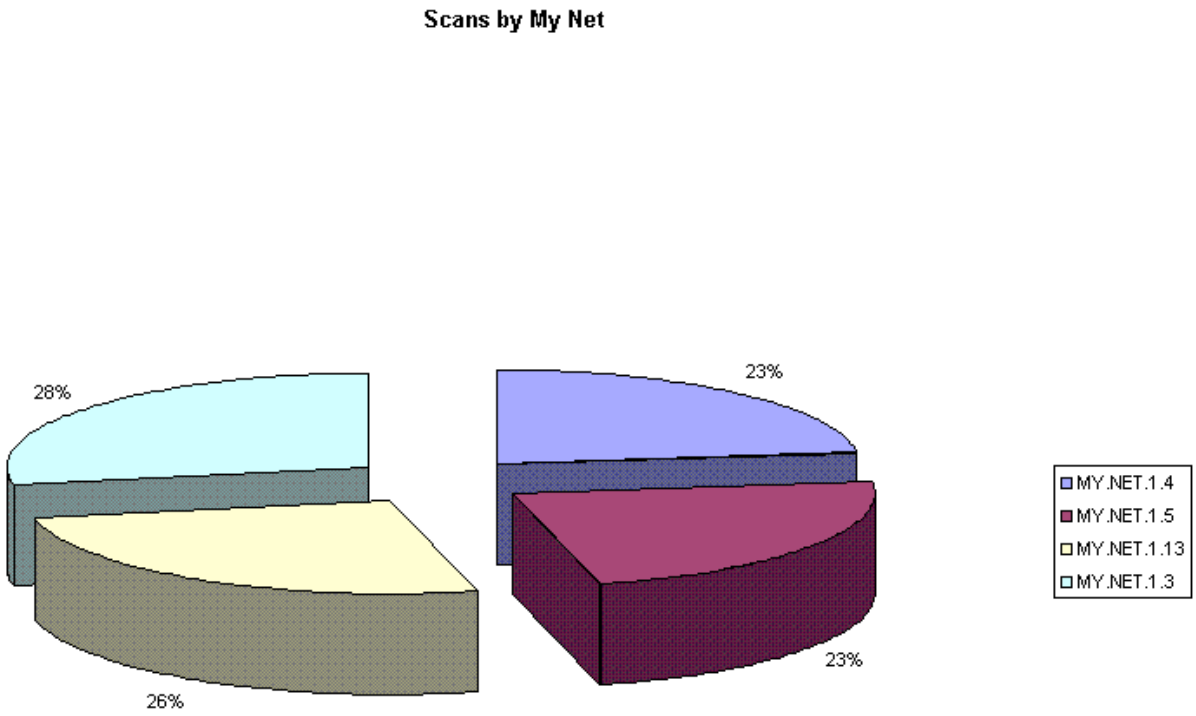
16:45:56.160854 arp who-has 192.168.1.40 tell 192.168.1.30
16:45:56.160973 arp reply 192.168.1.40 is-at 0:50:da:ca:4c:71

Base 64 encoding and Cast 256 encryption
In the test network we simulated several different tfn2k commands (-c3, -c4, -c5, -c6) and it appears that with different commands the encoding is somewhat different, therefore, theoretically, signatures could be made based on the different commands that are issued, which would help you determine what type of flood activity is occurring.

Section 3 Snort Data Analysis

Scenario: Your organization has been asked to provide a bid to provide security services for GIAC Enterprises, a dot.com startup that sells electronic fortune cookie sayings. You have been provided with data a Snort XXXX System with a fairly standard rulebase for a month. Not all the data has been collected for various reasons.

This section is an analyses of the data, keying in on the alerts, scans, network problems and possible compromises. Data of interest has been pasted into the analysis for clarity and correlation examples have also been provided to strengthen the analysis. I separated the amalgamated and sorted data into three folders - scans - alerts - data 1. Using parts of Lenny Zeltser 's scripts from the correlate.tar file (zeltser.com) correlated the snort scan data into a prioritized format. Insert graph and chart. A quick sort of the scan data using excel, sorting by column A and then by column B identified the scans from inside the my.net network. I was somewhat concerned about all the activity (scanning) that appeared to occur and therefore spent a considerable amount of time verifying what the stimulus was. It was particularly important to establish whether or not some outside stimulus had triggered the internal scanning activity or whether or not I was seeing only one side of the communication. As you will see below there are several scenarios to explain the scanning activity.



From the pie chart above 28% of the scans initiated from internal hosts were generated by MY.NET.1.3 followed by 26% from 1.13, 23% from 1.5 and 23% from 1.4.

Analysis of MY.NET.1.3

Aug 15 14:05:35 MY.NET.1.3:53 -> MY.NET.101.89:44771 UDP
Aug 15 14:05:35 MY.NET.1.3:53 -> MY.NET.101.89:44773 UDP
Aug 15 14:05:35 MY.NET.1.3:53 -> MY.NET.101.89:44774 UDP
Aug 16 15:44:15 MY.NET.1.3:53 -> MY.NET.101.89:53051 UDP
Aug 16 15:44:15 MY.NET.1.3:53 -> MY.NET.101.89:53052 UDP
Aug 16 15:44:15 MY.NET.1.3:53 -> MY.NET.101.89:53053 UDP
Aug 17 05:45:23 MY.NET.1.3:53 -> MY.NET.101.89:56921 UDP

Aug 17 05:45:23 MY.NET.1.3:53 -> MY.NET.101.89:56925 UDP
Aug 17 05:45:23 MY.NET.1.3:53 -> MY.NET.101.89:56928 UDP
Aug 18 08:16:36 MY.NET.1.3:53 -> MY.NET.101.89:33296 UDP
Aug 18 08:16:36 MY.NET.1.3:53 -> MY.NET.101.89:33297 UDP
Aug 18 08:16:36 MY.NET.1.3:53 -> MY.NET.101.89:33300 UDP
Aug 28 07:21:00 MY.NET.1.3:53 -> MY.NET.101.89:64932 UDP
Aug 28 07:21:01 MY.NET.1.3:53 -> MY.NET.101.89:64937 UDP
Aug 28 07:21:02 MY.NET.1.3:53 -> MY.NET.101.89:64943 UDP
Sep 11 09:43:31 MY.NET.1.3:53 -> MY.NET.101.89:35842 UDP
Sep 11 09:43:33 MY.NET.1.3:53 -> MY.NET.101.89:35844 UDP
Sep 11 09:43:33 MY.NET.1.3:53 -> MY.NET.101.89:35845 UDP

In Lenny Zeltser's(www.zeltser.com and www.sans.org) analysis he also keys in on the DNS traffic on MY.NET.1.3, wherein he suggests that "...MY.NET.1.3 to rank #1 among scan source hosts located on the MY.NET network. However, it was most likely a false positive, indicating that MY.NET.1.13 is a heavily loaded DNS server." Building upon Lenny's I evaluated what the other most active internal hosts were doing in terms of DNS. After cat 'ing and grep 'ing for DNS traffic on the other very active internal host, the MY.NET.1.4. I utilised the diff command to view the differences or similarities in this case between the traffic on MY.NET.1.3 and MY.NET.1.4.

Below are the commands for the searches, the diff command followed by a selection of the activities side by side for both the 1.4 and the 1.3 hosts separated by the ‘ | ‘

```
cat analysis.1.3.sorted | grep "Sep 3" > analsis.1.3.sorted.just.sep3
cat analysis.1.4.sorted | grep "Sep 3" > analsis.1.4.sorted.just.sep3

diff -iwbHy analsis.1.3.sorted.just.sep3 analsis.1.4.sorted.just.sep3 > diff.sep3
```

Sep 3 09:03:24 MY.NET.1.3:53 -> MY.NET.70.98:1517 UDP | Sep 3 09:03:24 MY.NET.1.4:53 -> MY.NET.53.91:1425 UDP
Sep 3 09:03:25 MY.NET.1.3:123 -> MY.NET.60.174:123 UDP | Sep 3 09:03:24 MY.NET.1.4:53 -> MY.NET.99.117:36138 UDP
Sep 3 09:03:25 MY.NET.1.3:53 -> MY.NET.53.151:2269 UDP | Sep 3 09:03:25 MY.NET.1.4:123 -> MY.NET.139.120:123 UDP
Sep 3 09:03:25 MY.NET.1.3:53 -> MY.NET.53.30:1578 UDP | Sep 3 09:03:25 MY.NET.1.4:53 -> MY.NET.100.152:1291 UDP
Sep 3 09:03:25 MY.NET.1.3:53 -> MY.NET.53.88:4378 UDP | Sep 3 09:03:25 MY.NET.1.4:53 -> MY.NET.120.32:1302 UDP
Sep 3 09:03:26 MY.NET.1.3:123 -> MY.NET.100.52:123 UDP | Sep 3 09:03:25 MY.NET.1.4:53 -> MY.NET.152.167:1055 UDP
Sep 3 09:03:26 MY.NET.1.3:53 -> MY.NET.179.86:1262 UDP | Sep 3 09:03:25 MY.NET.1.4:53 -> MY.NET.53.151:2269 UDP
Sep 3 09:03:26 MY.NET.1.3:53 -> MY.NET.5.203:1027 UDP | Sep 3 09:03:25 MY.NET.1.4:53 -> MY.NET.53.30:1578 UDP
Sep 3 09:03:26 MY.NET.1.3:53 -> MY.NET.53.219:2926 UDP | Sep 3 09:03:25 MY.NET.1.4:53 -> MY.NET.53.88:4378 UDP
Sep 3 09:03:26 MY.NET.1.3:53 -> MY.NET.70.38:2185 UDP | Sep 3 09:03:26 MY.NET.1.4:53 -> MY.NET.162.87:1240 UDP
Sep 3 09:03:26 MY.NET.1.3:53 -> MY.NET.70.98:1518 UDP | Sep 3 09:03:26 MY.NET.1.4:53 -> MY.NET.53.219:2926 UDP

Analysis of MY.NET.1.5
Using methods described above it appears that the 1.5 machine is also acting as a DNS for several machines inside the MY.NET network

Sep 3 09:03:18 MY.NET.1.3:53 -> MY.NET.100.148:2563 UDP | Sep 3 09:03:18 MY.NET.1.5:53 -> MY.NET.152.146:1953 UDP
Sep 3 09:03:18 MY.NET.1.3:53 -> MY.NET.152.146:1953 UDP | Sep 3 09:03:18 MY.NET.1.5:53 -> MY.NET.179.78:2082 UDP
Sep 3 09:03:19 MY.NET.1.3:53 -> MY.NET.111.166:1509 UDP | Sep 3 09:03:19 MY.NET.1.5:53 -> MY.NET.152.55:1387 UDP
Sep 3 09:03:19 MY.NET.1.3:53 -> MY.NET.111.18:1856 UDP | Sep 3 09:03:19 MY.NET.1.5:53 -> MY.NET.179.52:1807 UDP
Sep 3 09:03:19 MY.NET.1.3:53 -> MY.NET.152.55:1387 UDP | Sep 3 09:03:19 MY.NET.1.5:53 -> MY.NET.179.78:2085 UDP
Sep 3 09:03:19 MY.NET.1.3:53 -> MY.NET.162.198:4363 UDP | Sep 3 09:03:19 MY.NET.1.5:53 -> MY.NET.53.180:3615 UDP
Sep 3 09:03:19 MY.NET.1.3:53 -> MY.NET.53.180:3615 UDP | Sep 3 09:03:19 MY.NET.1.5:53 -> MY.NET.53.224:1107 UDP
Sep 3 09:03:19 MY.NET.1.3:53 -> MY.NET.53.219:2926 UDP | Sep 3 09:03:20 MY.NET.1.5:123 -> MY.NET.179.54:123 UDP
Sep 3 09:03:19 MY.NET.1.3:53 -> MY.NET.53.224:1107 UDP | Sep 3 09:03:20 MY.NET.1.5:53 -> MY.NET.99.46:38474 UDP
Sep 3 09:03:19 MY.NET.1.3:53 -> MY.NET.99.117:36136 UDP | Sep 3 09:03:21 MY.NET.1.5:123 -> MY.NET.60.161:123 UDP
Sep 3 09:03:20 MY.NET.1.3:53 -> MY.NET.110.159:1034 UDP | Sep 3 09:03:21 MY.NET.1.5:53 -> MY.NET.106.146:4581 UDP
Sep 3 09:03:20 MY.NET.1.3:53 -> MY.NET.152.167:1053 UDP | Sep 3 09:03:21 MY.NET.1.5:53 -> MY.NET.152.15:1986 UDP
Sep 3 09:03:20 MY.NET.1.3:53 -> MY.NET.5.203:1024 UDP | Sep 3 09:03:21 MY.NET.1.5:53 -> MY.NET.53.147:4499 UDP
Sep 3 09:03:21 MY.NET.1.3:53 -> MY.NET.106.146:4581 UDP | Sep 3 09:03:21 MY.NET.1.5:53 -> MY.NET.70.38:2185 UDP
Sep 3 09:03:21 MY.NET.1.3:53 -> MY.NET.152.15:1986 UDP | Sep 3 09:03:21 MY.NET.1.5:53 -> MY.NET.70.98:1516 UDP
Sep 3 09:03:21 MY.NET.1.3:53 -> MY.NET.152.159:1523 UDP | Sep 3 09:03:21 MY.NET.1.5:53 -> MY.NET.70.98:1517 UDP
Sep 3 09:03:21 MY.NET.1.3:53 -> MY.NET.163.30:2224 UDP | Sep 3 09:03:22 MY.NET.1.5:123 -> MY.NET.152.22:123 UDP

Analysis of the MY.NET.1.13
The 1.13 host has been the recipient of several different scans ranging from trojans to wingate as evident below

Aug 15 00:46:12 195.114.226.41:2255 -> MY.NET.1.13:21 SYN **S*****
Aug 16 04:35:21 35.10.82.111:2821 -> MY.NET.1.13:27374 SYN **S*****
Aug 16 04:58:52 35.10.82.111:2257 -> MY.NET.1.13:27374 SYN **S*****
Sep 2 14:16:12 194.165.230.250:1648 -> MY.NET.1.13:21 SYN **S*****
Sep 11 18:40:38 168.187.26.157:1522 -> MY.NET.1.13:1080 SYN **S*****

However it should be noted that there is a large amount of activity on the MY.NET.1.13 host pertaining to ports 7000, 7001, 7002, 7003, 7028. This caught my eye because an exterior host appeared to be very interested in port 7003 on the 1.13 machine. This activity is seen on several days, until most likely between August 18 and September 03 something malicious may have happened. However due to lack of technical data it is difficult to accurately say what happened.

Aug 15 01:25:08 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP
Aug 15 05:27:19 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP
Aug 15 06:27:20 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP
Aug 16 06:29:24 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP
Aug 16 15:30:43 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP
Aug 16 16:30:44 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP
Aug 16 06:29:24 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP
Aug 16 15:30:43 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP
Aug 16 16:30:44 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP
Aug 17 03:31:55 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP
Aug 17 04:31:56 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP
Aug 17 06:32:28 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP
Aug 18 06:35:27 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP
Aug 18 07:35:28 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP
Aug 18 22:36:27 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP

On September 3 the MY.NET.1.13 begins sending out UDP packets all over the internal network, UDP packets are from port 7003 -> 7001.

Sep 3 09:03:19 MY.NET.1.13:7003 -> MY.NET.100.83:7001 UDP
Sep 3 09:03:19 MY.NET.1.13:7003 -> MY.NET.110.82:7001 UDP
Sep 3 09:03:19 MY.NET.1.13:7003 -> MY.NET.60.164:7001 UDP
Sep 3 09:03:22 MY.NET.1.13:7003 -> MY.NET.53.106:7001 UDP
Sep 3 09:03:22 MY.NET.1.13:7003 -> MY.NET.53.207:7001 UDP
Sep 3 09:10:27 MY.NET.1.13:7003 -> MY.NET.60.9:7001 UDP
Sep 3 09:10:28 MY.NET.1.13:7003 -> MY.NET.152.214:7001 UDP
Sep 3 09:10:28 MY.NET.1.13:7003 -> MY.NET.60.202:7001 UDP

There are also connections to and from different ports in amongst the packets mentioned above, which need further examination and do not appear to be related to the Andrew File Server. My suggestion is to dump a couple of UDP packets and inspect their contents. This is recommended in spite of the fact that University associated with the 24.3.39.44 also uses AFS. There have been several mentions of the possibility that there is a AFS trojan operating on port 7007.

Correlations for this traffic were found by searching other practicals were correlations were found from Richard Salisko's practical which stated "…from July 29th to August 10th there is quite a bit of activity from host 24.3.39.44 on udp port 7001 to eight hosts on the internal networks. It appears to be legitimate traffic (probably NFS related) however it does bear checking into." As well a correlation was found for a possible Trojan operating on port 7000 in William Lorimer's practical were he identified port 7000 as possibly having a remote grab Trojan.

Jul 29 21:17:07 24.3.39.44:7001 -> MY.NET.6.42:7000 UDP

Jul 29 21:17:08 24.3.39.44:7001 -> MY.NET.70.142:7000 UDP

Jul 29 21:17:08 24.3.39.44:7001 -> MY.NET.60.43:7000 UDP

Jul 29 21:17:08 24.3.39.44:7001 -> MY.NET.6.45:7000 UDP

Jul 29 21:17:08 24.3.39.44:7001 -> MY.NET.6.33:7003 UDP

Jul 29 21:17:08 24.3.39.44:7001 -> MY.NET.6.48:7000 UDP

Jul 29 21:17:08 24.3.39.44:7001 -> MY.NET.60.12:7003 UDP

Jul 29 21:17:08 24.3.39.44:7001 -> MY.NET.1.13:7003 UDP

The most informative correlation came from the Joseph R RACH practical were several different scans that were believed to be generated by nmap showed up in his snort data. As you will note with the Andrew File System version 3 there are several different ports that are associated with the AFS. The difference in the detect below is that the originating port is associated with an nmap scan and the detects for MY.NET.1.13 are associated with a known afs port. Other than the fact that there may be a Trojan operating on some of the afs ports I think that there is something going on with the afs and is worth taking a look at as it seems to be generating a lot of traffic.

22:59:13.103305 SCANNER.OTHER.NET.43645 > WORKSTATION-01.MY.NET.afs3-errors: S 3162278929:3162278929(0) win 1024 (ttl 48, id 48055)

22:59:13.121919 WORKSTATION-01.MY.NET.afs3-errors > SCANNER.OTHER.NET.43645: R 0:0(0) ack 3162278930 win 0 (ttl 64, id 58194)

22:59:13.211228 SCANNER.OTHER.NET.43645 > WORKSTATION-01.MY.NET.afs3-rmtsys: S 3162278929:3162278929(0) win 1024 (ttl 48, id 11352)

22:59:13.212069 WORKSTATION-01.MY.NET.afs3-rmtsys > SCANNER.OTHER.NET.43645: R 0:0(0) ack 3162278930 win 0 (ttl 64, id 45058)

22:59:13.442209 SCANNER.OTHER.NET.43645 > WORKSTATION-01.MY.NET.afs3-bos: S 3162278929:3162278929(0) win 1024 (ttl 48, id 51606)

22:59:13.443642 WORKSTATION-01.MY.NET.afs3-bos > SCANNER.OTHER.NET.43645: R 0:0(0) ack 3162278930 win 0 (ttl 64, id 43155)

22:59:13.640364 SCANNER.OTHER.NET.43645 > WORKSTATION-01.MY.NET.afs: S 3162278929:3162278929(0) win 1024 (ttl 48, id 23895)

22:59:14.667565 SCANNER.OTHER.NET.43645 > WORKSTATION-01.MY.NET.afs3-fileserver: S 3162278929:3162278929(0) win 1024 (ttl 48, id 44452)

22:59:14.683975 WORKSTATION-01.MY.NET.afs3-fileserver > SCANNER.OTHER.NET.43645: R 0:0(0) ack 3162278930 win 0 (ttl 64, id 43820)

22:59:15.123158 SCANNER.OTHER.NET.43645 > WORKSTATION-01.MY.NET.afs3-callback: S 3162278929:3162278929(0) win 1024 (ttl 48, id 4527)

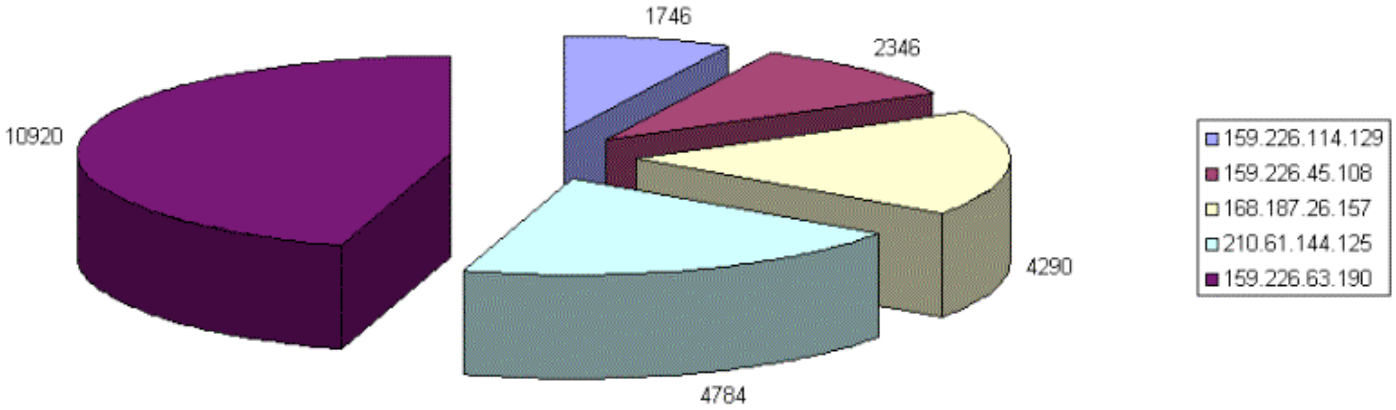
22:59:15.125098 SCANNER.OTHER.NET.43645 > WORKSTATION-01.MY.NET.afs3-volser: S 3162278929:3162278929(0) win 1024 (ttl 48, id 33267)

22:59:15.129961 WORKSTATION-01.MY.NET.afs3-callback > SCANNER.OTHER.NET.43645: R 0:0(0) ack 3162278930 win 0 (ttl 64, id 46203)

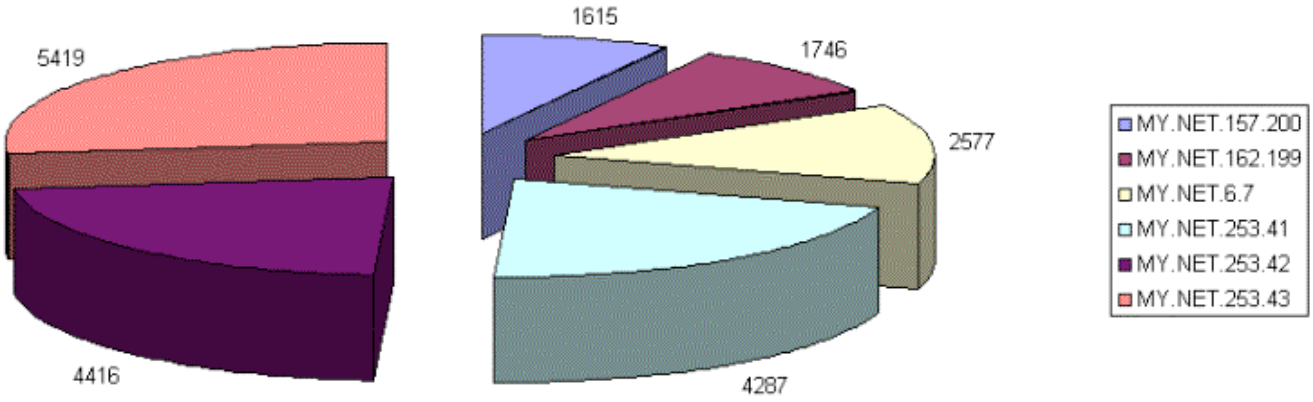
22:59:15.134081 WORKSTATION-01.MY.NET.afs3-volser > SCANNER.OTHER.NET.43645: R 0:0(0) ack 3162278930 win 0 (ttl 64, id 38806)

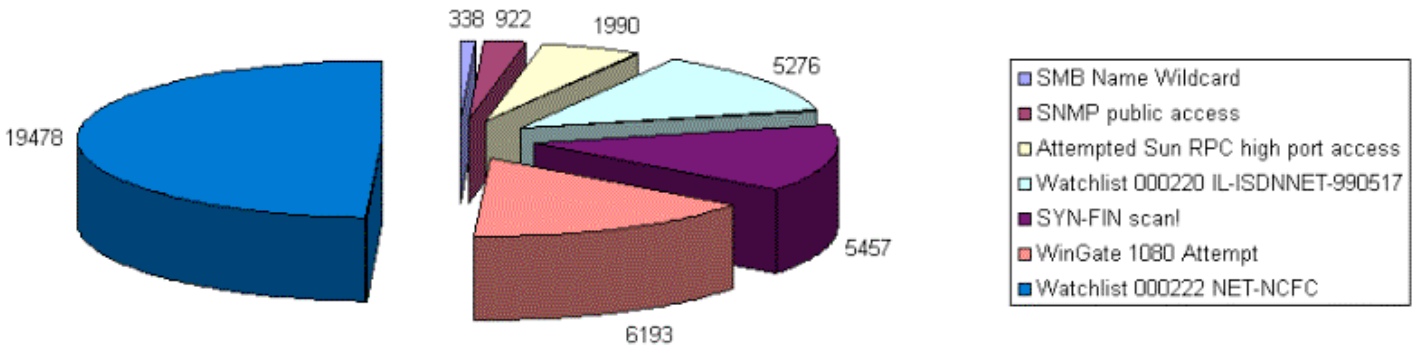
22:59:15.187344 SCANNER.OTHER.NET.43645 > WORKSTATION-01.MY.NET.afs3-kaserver: S 3162278929:3162278929(0) win 1024 (ttl 48, id 65103)

Alerts by Source



Alerts by Destination





19478 Alerts generated by Watchlist 000222
6193 Alerts generated by Wingate 1080 Attempts
5457 Alerts generated by SYN-FIN scans
5276 Alerts generated by Watchlist 000220
1990 Alerts Generated by Attempted Sun RPC high port access

--*-*-*-*

Some of the Alerts generated by the Chinese Academy of Sciences (CAS) are believed to be legitimate such as the email and the telnet traffic. It is difficult to determine with certainty that the traffic is legitimate.

Some items of interest arising from the CAS alerts is that it appears as though CAS is scanning for the RAT trojan but we also have to take into account that it is possible that the rule file is only picking up CAS connections coming into MY.NET
if we had a complete dataset we may find that MY.NET actually initiated the ftp transactions. I suggest pulling one of the packets and seeing what is in the contents for two reasons
1. to assure yourself that it is legitimate ftp traffic
2. there is not allot of information available on the RAT trojan and it could be worthy to the ID community of someone could figure it out.

08/20-15:12:47.329322 [**] Watchlist 000222 NET-NCFC [**] 159.226.114.129:21 -> MY.NET.162.199:1095
08/20-15:12:48.361943 [**] Watchlist 000222 NET-NCFC [**] 159.226.114.129:21 -> MY.NET.162.199:1095
08/20-15:12:48.834972 [**] Watchlist 000222 NET-NCFC [**] 159.226.114.129:21 -> MY.NET.162.199:1095
08/20-15:12:49.029066 [**] Watchlist 000222 NET-NCFC [**] 159.226.114.129:21 -> MY.NET.162.199:1095

08/20-15:12:56.375558 [**] Watchlist 000222 NET-NCFC [**] 159.226.114.129:37266 -> MY.NET.162.199:1096
08/20-15:13:06.988956 [**] Watchlist 000222 NET-NCFC [**] 159.226.114.129:37268 -> MY.NET.162.199:1097
08/20-15:13:08.993543 [**] Watchlist 000222 NET-NCFC [**] 159.226.114.129:37268 -> MY.NET.162.199:1097

08/20-15:13:09.409396 [**] Watchlist 000222 NET-NCFC [**] 159.226.114.129:21 -> MY.NET.162.199:1095
08/20-15:13:09.453578 [**] Watchlist 000222 NET-NCFC [**] 159.226.114.129:37268 -> MY.NET.162.199:1097
08/20-15:17:43.161253 [**] Watchlist 000222 NET-NCFC [**] 159.226.114.129:37270 -> MY.NET.162.199:1098
08/20-15:17:43.209009 [**] Watchlist 000222 NET-NCFC [**] 159.226.114.129:37270 -> MY.NET.162.199:1098
08/20-15:17:43.209617 [**] Watchlist 000222 NET-NCFC [**] 159.226.114.129:21 -> MY.NET.162.199:1095
08/20-15:17:43.383063 [**] Watchlist 000222 NET-NCFC [**] 159.226.114.129:37270 -> MY.NET.162.199:1098

*_*_*_*_*_*

The wingate attempts are probably looking for default wingate installs with easy password or default password so they can divert their traffic. Here is a snippet of the 1080 alerts and some of the most prolific offender, 168.182.26.157 which resolves to the Kuwaiti Ministry of Communications. Wingate exploits are looking for proxy servers 1080 probes (IDS 175). Correlations to this activity can be found at www.sans.org/y2k/060700.htm and www.sans.org/y2k/030200.htm.

09/11-18:40:36.435240 [**] WinGate 1080 Attempt [**] 168.187.26.157:1518 -> MY.NET.1.9:1080
09/11-18:40:36.435339 [**] WinGate 1080 Attempt [**] 168.187.26.157:1519 -> MY.NET.1.10:1080
09/11-18:40:36.455258 [**] WinGate 1080 Attempt [**] 168.187.26.157:1520 -> MY.NET.1.11:1080
...
...
...
09/11-19:15:59.677638 [**] WinGate 1080 Attempt [**] 168.187.26.157:1301 -> MY.NET.54.251:1080
09/11-19:16:00.738628 [**] WinGate 1080 Attempt [**] 168.187.26.157:1271 -> MY.NET.54.235:1080
09/11-19:16:00.783231 [**] WinGate 1080 Attempt [**] 168.187.26.157:1270 -> MY.NET.54.234:1080

The CVE entries for the above mentioned scans can be found at http://cve.mitre.org/ CVE-1999-0290, CVE-1999-0291, CVE-1999-0441, CVE-1999-0494, CAN-1999-0657

*_*_*_*_*_*

SYN-FIN scans are tipping the scale at 5457, these types of scans do not occur naturally and are evidence of packet crafting, there has been an increase in these types of scans.

Karen Frederick has put together an extensive look at the 9704 -> 9704 activity which caused me to go back and take a second look at the activity. Karen's Practical can be found
at sans.org/y2k/analysts.htm.

09/07-21:33:23.187413 [**] SYN-FIN scan! [**] 213.25.136.60:9704 -> MY.NET.1.4:9704
09/07-21:33:23.267012 [**] SYN-FIN scan! [**] 213.25.136.60:9704 -> MY.NET.1.8:9704
09/07-21:33:23.749708 [**] SYN-FIN scan! [**] 213.25.136.60:9704 -> MY.NET.1.32:9704

and from the snort data file related to the above trace

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
09/07-21:33:30.514289 213.25.136.60:9704 -> MY.NET.1.4:9704
TCP TTL:29 TOS:0x0 ID:39426
**SF**** Seq: 0x215466F8   Ack: 0x36F7C1CA   Win: 0x404
05 FF 20 20 8E 64                                 ..  .d

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
09/07-21:33:30.594266 213.25.136.60:9704 -> MY.NET.1.8:9704
TCP TTL:29 TOS:0x0 ID:39426
**SF**** Seq: 0x215466F8   Ack: 0x36F7C1CA   Win: 0x404
CF 45 40 84 C6 99                                 .E@...

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
09/07-21:33:30.633739 213.25.136.60:9704 -> MY.NET.1.10:9704
TCP TTL:29 TOS:0x0 ID:39426
**SF**** Seq: 0x215466F8   Ack: 0x36F7C1CA   Win: 0x404
63 FB 70 A0 5D 27                                 c.p.]'

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Analysing the files together we see strong linkages that were discussed in Karen Frederick's practical:

“The correlations seems to indicate that this attack has a very distinct signature. All of the packets in the correlations have source port 9704, destination port 9704, and the SYN and FIN flags set. Also, three of the correlations show packet ID numbers; in all cases, the packet IDs are set to 39426. In many cases, the scans to port 9704 are preceded by similar scans to port 1524; in some cases, port 2222 is scanned after port 1524 but before port 9704.”

And a further correlation is found at (http://www.sans.org/y2k/091100.htm)

Sep 11 05:01:00 seeker snort[5149]: spp_portscan: End of portscan from

216.242.88.30: TOTAL time(0s) hosts(1) TCP(1) UDP(0) STEALTH

Here is an Hex dump of the packet intercepted on the 13 Sep:

asctcpdump -s 1518 -x ip and host 213.25.136.60 -n -v -r tcp.2000091318

18:34:42.229142 213.25.136.60.9704 > 192.168.30.1.9704: SF

729879104:729879104(0) win 1028 (ttl 25, id 39426)

4500 0028 9a02 0000 1906 0d15 d519 883c E..(.....<

xxxx xxxx 25e8 25e8 2b81 1240 4220 1369 .p..%.%.+..@B .i

5003 0404 d209 0000 0000 0000 0000

P.....

Snort intercept - Sep 13, 2000

Sep 13 18:34:42 seeker snort[13345]: spp_portscan:

PORTSCAN DETECTED from 213.25.136.60 (STEALTH)

Sep 13 18:34:42 seeker snort[13345]: SCAN-SYN FIN:

213.25.136.60:9704 -> 192.168.30.1:9704

Sep 13 19:00:04 seeker snort[13345]: spp_portscan:

portscan status from 213.25.136.60: 1 connections across 1 hosts:

TCP(1), UDP(0) STEALTH

Sep 13 19:00:17 seeker snort[13345]: spp_portscan:

End of portscan from 213.25.136.60: TOTAL time(0s) hosts(1)

TCP(1) UDP(0) STEALTH

Further information on the 9704 port scanning can be found at:
http://cve.mitre.org/cgi-bin CAN-2000-0666.

http://securityportal.com/topnews/weekly/solaris20000821.html

http://www.securityfocus.com/bid/1480

While just the fact that SYN-FIN packets do not occur normally, it should also be noted that the SEQ ID is not changing as it should, it continues on for 9 repetitions (see below the alert file for the snort data)

09/11-07:06:43.022244 [**] SYN-FIN scan! [**] 210.61.144.125:21 -> MY.NET.253.214:21
09/11-07:06:44.215607 [**] SYN-FIN scan! [**] 210.61.144.125:21 -> MY.NET.254.18:21
09/11-07:06:44.737582 [**] SYN-FIN scan! [**] 210.61.144.125:21 -> MY.NET.254.45:21

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
09/11-07:06:43.233340 210.61.144.125:21 -> MY.NET.253.129:21
TCP TTL:25 TOS:0x0 ID:39426
**SF**** Seq: 0x3AD56C91   Ack: 0x464A253B   Win: 0x404
00 00 00 00 00 00                               ......
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
09/11-07:06:43.292982 210.61.144.125:21 -> MY.NET.253.132:21
TCP TTL:25 TOS:0x0 ID:39426
**SF**** Seq: 0x3AD56C91   Ack: 0x464A253B   Win: 0x404
00 00 00 00 00 00                               ......
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
09/11-07:06:43.572364 210.61.144.125:21 -> MY.NET.253.146:21
TCP TTL:25 TOS:0x0 ID:39426
**SF**** Seq: 0x3AD56C91   Ack: 0x464A253B   Win: 0x404
00 00 00 00 00 00                               ......
```

Ack for the above mentioned snort data for the syn-fin activity is 1179264315 and the Seq is 987065489

*_*_*_*_*_*
5276 Alerts from Israel ISD Net with the majority being:

09/03-03:37:20.882995 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.27.111:1526 -> MY.NET.206.154:6700
09/03-03:37:28.204496 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.27.111:1526 -> MY.NET.206.154:6700
09/03-03:37:30.893442 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.27.111:1526 -> MY.NET.206.154:6700

There were several other correlations to data on MY.NET hosts being contacted by exterior hosts via 6346 such as
09/06 20:15:16 128.148.221.23:6346 -> MY.NET.223.62:4352. Therefore I suggest taking a look at the packets to see what is going on. A search of google.com revelaed that this traffic is gnutella

09/06-19:00:57.513744 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.47.30:6346 -> MY.NET.223.62:2995
09/06-19:00:57.629395 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.47.30:6346 -> MY.NET.223.62:2995
09/06-19:01:16.598655 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.47.30:6346 -> MY.NET.223.62:2995

Possibly some mail traffic, however without the full packet there is no way to tell, worth taking a second look at. A search of the data file for 212.179 revealed no hits.
09/07-03:27:11.855673 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.58.4:42790 -> MY.NET.253.42:25
09/07-03:27:14.459009 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.58.4:42795 -> MY.NET.253.41:25
09/07-03:27:29.948302 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.58.4:42795 -> MY.NET.253.41:25

Port 6699 is napster traffic

09/09-10:45:10.461542 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.66.2:22756 -> MY.NET.221.94:6699
09/09-10:45:14.732549 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.66.2:22756 -> MY.NET.221.94:6699
09/09-10:45:15.340444 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.66.2:22756 -> MY.NET.221.94:6699

09/12-10:20:43.120997 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.127.45:1063 -> MY.NET.202.58:6688
09/12-10:20:44.599975 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.127.45:1063 -> MY.NET.202.58:6688
09/12-10:20:45.579690 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.127.45:1063 -> MY.NET.202.58:6688

Port 6699 is napster traffic
09/13-12:10:54.004320 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.58.204:1430 -> MY.NET.205.254:6699

Could be gnutella traffic
09/13-15:09:12.472237 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.61.5:21263 -> MY.NET.204.150:2669
09/13-15:09:22.292419 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.61.5:21263 -> MY.NET.204.150:2669
09/13-15:09:25.294601 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.61.5:21263 -> MY.NET.204.150:2669

Port 6699 is napster traffic
09/14-07:41:19.868471 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.58.174:2172 -> MY.NET.157.200:6699
09/14-07:41:20.058603 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.58.174:2172 -> MY.NET.157.200:6699
09/14-07:41:23.197469 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.58.174:2172 -> MY.NET.157.200:6699

Might want to take a look at this one as someone is surfing to secure http port. Worth taking a look to see what is exactly going on, especially since you have this IP on the watchlist
08/18-07:09:45.681474 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.61.252:1875 -> MY.NET.5.29:443
08/18-07:09:47.333786 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.61.252:1875 -> MY.NET.5.29:443
08/18-07:09:47.736461 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.61.252:1875 -> MY.NET.5.29:443

There is one mention of a Trojan operating on 4407 www.sans.org/y2k/010100-945.htm but there is not enough discernable information to suggest that your machine was indeed infected. It may be in your interest to tak a look at the contents of future packets destined for 4407. In the meantime a good virus scanner should detect the virus if it is in the wild.
08/20-09:05:41.982563 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.29.150:1098 -> MY.NET.53.28:4407
08/20-09:05:42.671792 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.29.150:1098 -> MY.NET.53.28:4407
08/20-09:05:44.418409 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.29.150:1098 -> MY.NET.53.28:4407

08/20-10:22:34.092107 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.61.244:2350 -> MY.NET.5.29:443
08/20-10:22:34.291756 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.61.244:2350 -> MY.NET.5.29:443
08/20-10:22:36.744553 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.61.244:2350 -> MY.NET.5.29:443

RPC alerts or false positives ?

As discussed by Lenny Zeltser in his practical (zeltser.com) the IP 205.188.53.98 is an AOL ICQ client and is probably legitimate ICQ traffic and not an attempted exploit of the 32771(ruserd). for as Lenny puts it
"...such vulnerabilities are typically exploits via buffer overflows, which do not require several days to work."
Correlations of this traffic can be found sans.org/y2k/060100.htm

09/05-20:50:20.124678 [**] Attempted Sun RPC high port access [**] 205.188.153.98:4000 -> MY.NET.222.66:32771
09/05-20:50:22.123597 [**] Attempted Sun RPC high port access [**] 205.188.153.98:4000 -> MY.NET.222.66:32771
09/05-20:50:25.125245 [**] Attempted Sun RPC high port access [**] 205.188.153.98:4000 -> MY.NET.222.66:32771
...
...
...
09/09-23:29:05.569902 [**] Attempted Sun RPC high port access [**] 205.188.153.98:4000 -> MY.NET.217.82:32771
09/09-23:30:05.480270 [**] Attempted Sun RPC high port access [**] 205.188.153.98:4000 -> MY.NET.217.82:32771
09/09-23:36:05.151117 [**] Attempted Sun RPC high port access [**] 205.188.153.98:4000 -> MY.NET.217.82:32771

Other hosts that are apparent ICQ clients which generated 1987 alerts
09/02-00:11:19.591313 [**] Attempted Sun RPC high port access [**] 205.188.153.109:4000 -> MY.NET.219.26:32771
09/03-18:14:45.936781 [**] Attempted Sun RPC high port access [**] 205.188.153.114:4000 -> MY.NET.206.38:32771
09/05-20:50:20.124678 [**] Attempted Sun RPC high port access [**] 205.188.153.98:4000 -> MY.NET.222.66:32771
09/07-06:56:21.278582 [**] Attempted Sun RPC high port access [**] 205.188.153.98:4000 -> MY.NET.220.58:32771
09/08-09:58:19.451527 [**] Attempted Sun RPC high port access [**] 205.188.153.112:4000 -> MY.NET.105.2:32771
09/09-07:48:01.881634 [**] Attempted Sun RPC high port access [**] 205.188.153.98:4000 -> MY.NET.217.82:32771
09/09-09:10:26.847986 [**] Attempted Sun RPC high port access [**] 205.188.153.115:4000 -> MY.NET.53.15:32771
09/11-13:03:28.791669 [**] Attempted Sun RPC high port access [**] 205.188.153.115:4000 -> MY.NET.218.218:32771
08/15-04:21:40.049280 [**] Attempted Sun RPC high port access [**] 205.188.179.33:4000 -> MY.NET.217.42:32771

Miscellaneous Alerts and Analysis

Tiny Fragments
These packets are an attempt to evade an IDS however the client will rebuild the tiny packets which most likely will lead to a malicious

event. A common program to do this is fragrouter.

09/11-13:20:45.833385 [**] Tiny Fragments - Possible Hostile Activity [**] 24.68.58.96 -> MY.NET.217.82
09/11-13:21:13.480527 [**] Tiny Fragments - Possible Hostile Activity [**] 24.68.58.96 -> MY.NET.217.82
09/11-13:20:45.833385 [**] Tiny Fragments - Possible Hostile Activity [**] 24.68.58.96 -> MY.NET.217.82
09/11-13:21:13.480527 [**] Tiny Fragments - Possible Hostile Activity [**] 24.68.58.96 -> MY.NET.217.82
09/14-09:15:35.433598 [**] Tiny Fragments - Possible Hostile Activity [**] 62.76.42.18 -> MY.NET.1.8
09/14-09:15:35.939752 [**] Tiny Fragments - Possible Hostile Activity [**] 62.76.42.18 -> MY.NET.1.9
09/14-09:15:44.375639 [**] Tiny Fragments - Possible Hostile Activity [**] 62.76.42.18 -> MY.NET.1.10
09/14-19:15:29.350830 [**] Tiny Fragments - Possible Hostile Activity [**] 62.76.42.17 -> MY.NET.212.86

It should be noted that 62.76.42.18 only hit the radar with tiny fragments and that the IP is associated with the Samara State Aerospace University and specifically the Diffractive Optics Laboratory. You may want to take a look at the hosts that received these tiny fragments.

SNMP Alerts

Should probably change the community string as you could be leaking too much information in relation to your network architecture to anyone who queries

09/02-07:54:01.499976 [**] SNMP public access [**] MY.NET.98.181:1051 -> MY.NET.101.192:161
09/02-07:54:03.197017 [**] SNMP public access [**] MY.NET.98.181:1052 -> MY.NET.101.192:161
09/02-07:54:08.590997 [**] SNMP public access [**] MY.NET.98.181:1055 -> MY.NET.101.192:161
...
...
...
08/19-11:55:29.929081 [**] SNMP public access [**] MY.NET.98.148:1113 -> MY.NET.101.192:161
08/19-11:55:29.946206 [**] SNMP public access [**] MY.NET.98.148:1116 -> MY.NET.101.192:161
08/19-11:55:30.142169 [**] SNMP public access [**] MY.NET.98.148:1117 -> MY.NET.101.192:161

SMB Wildcard Access

09/02-07:54:03.629167 [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137
09/02-07:56:25.440516 [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137
09/02-07:58:49.086208 [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137

09/09-10:52:16.881113 [**] SMB Name Wildcard [**] 129.37.160.81:137 -> MY.NET.100.130:137
08/18-12:13:28.981943 [**] SMB Name Wildcard [**] 4.17.88.66:137 -> MY.NET.6.15:137
08/18-12:13:30.463668 [**] SMB Name Wildcard [**] 4.17.88.66:137 -> MY.NET.6.15:137

I am not sure how much of this traffic you want over the internet. WINS or drive mapping. I am okay with the dialups from the campus such as umcp-222.dialup.umbc (131.118.254.222) however the dialup mapping that is going on with modem-18.ecthelion.dialup.pol.co.uk is somewhat concerning. You also have some mapping from nga.gov which is from the national art gallery, which could be legitimate. I have included all the 137 traffic as I am concerned about sharing 137 over the internet.

62.136.168.18 is a dialup in the UK
207.79.66.3 is the National Gallery of Art
131.118.254.222 is a dialup associated with umbc
205.229.90.194 is a dialup from senior campus living on umbc

08/19-22:57:41.121096 [**] SMB Name Wildcard [**] 129.37.161.200:137 -> MY.NET.100.130:137
08/19-22:58:05.400374 [**] SMB Name Wildcard [**] 129.37.161.200:137 -> MY.NET.100.130:137
08/19-22:58:39.817360 [**] SMB Name Wildcard [**] 129.37.161.200:137 -> MY.NET.100.130:137
08/19-22:58:42.825934 [**] SMB Name Wildcard [**] 129.37.161.200:137 -> MY.NET.100.130:137
08/11-16:13:19.788046 [**] SMB Name Wildcard [**] 205.229.90.194:63733 -> MY.NET.181.37:137
08/11-16:15:33.790499 [**] SMB Name Wildcard [**] 131.118.254.222:137 -> MY.NET.6.7:137
08/11-16:15:35.294638 [**] SMB Name Wildcard [**] 131.118.254.222:137 -> MY.NET.6.7:137
08/11-16:15:36.800020 [**] SMB Name Wildcard [**] 131.118.254.222:137 -> MY.NET.6.7:137
08/11-16:16:04.289653 [**] SMB Name Wildcard [**] 168.143.29.9:137 -> MY.NET.60.17:137
08/11-16:16:05.792099 [**] SMB Name Wildcard [**] 168.143.29.9:137 -> MY.NET.60.17:137
08/11-16:26:04.299200 [**] SMB Name Wildcard [**] 62.136.168.18:1124 -> MY.NET.70.121:137
08/11-16:26:05.696669 [**] SMB Name Wildcard [**] 168.143.29.9:137 -> MY.NET.60.17:137
08/11-16:26:23.147585 [**] SMB Name Wildcard [**] 62.136.168.18:1124 -> MY.NET.70.121:137

08/11-16:28:02.117131 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.230:137
08/11-16:28:03.736505 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.230:137
08/11-16:28:05.108614 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.230:137
08/11-16:28:46.525879 [**] SMB Name Wildcard [**] 207.79.66.3:614 -> MY.NET.253.53:137
08/11-16:28:46.525941 [**] SMB Name Wildcard [**] 207.79.66.3:137 -> MY.NET.253.53:137
08/11-16:28:48.020537 [**] SMB Name Wildcard [**] 207.79.66.3:137 -> MY.NET.253.53:137
08/11-16:28:49.521266 [**] SMB Name Wildcard [**] 207.79.66.3:614 -> MY.NET.253.53:137
08/11-16:28:49.522863 [**] SMB Name Wildcard [**] 207.79.66.3:137 -> MY.NET.253.53:137
08/11-16:30:34.358284 [**] SMB Name Wildcard [**] 131.118.254.222:137 -> MY.NET.6.7:137
08/11-16:30:37.362453 [**] SMB Name Wildcard [**] 131.118.254.222:137 -> MY.NET.6.7:137
08/11-16:34:17.629689 [**] SMB Name Wildcard [**] 162.33.184.239:137 -> MY.NET.60.11:137
08/11-16:34:19.119174 [**] SMB Name Wildcard [**] 162.33.184.239:137 -> MY.NET.60.11:137
08/11-16:34:20.629867 [**] SMB Name Wildcard [**] 162.33.184.239:137 -> MY.NET.60.11:137
08/11-16:35:13.544620 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.165:137
08/11-16:35:27.109652 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.165:137
08/11-16:35:28.599806 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.165:137
08/11-16:35:33.143680 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.165:137
08/11-16:36:08.602748 [**] SMB Name Wildcard [**] 168.143.29.9:137 -> MY.NET.60.17:137
08/11-16:37:10.502024 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.165:137
08/11-16:37:11.949276 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.165:137
08/11-16:37:13.825394 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.165:137
08/11-16:37:13.829483 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.165:137
08/11-16:38:04.318402 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.165:137
08/11-16:38:40.624689 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.165:137
08/11-16:38:41.032245 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.165:137
08/11-16:38:42.143327 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.165:137
08/11-16:39:35.261511 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.230:137
08/11-16:40:25.799139 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.230:137
08/11-16:41:14.018922 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.230:137
08/11-16:41:20.567074 [**] SMB Name Wildcard [**] 216.164.133.254:137 -> MY.NET.60.8:137
08/11-16:42:10.039852 [**] SMB Name Wildcard [**] 206.171.108.1:724 -> MY.NET.6.7:137
08/11-16:42:11.495774 [**] SMB Name Wildcard [**] 206.171.108.1:724 -> MY.NET.6.7:137
08/11-16:45:37.056832 [**] SMB Name Wildcard [**] 131.118.254.222:137 -> MY.NET.6.7:137
08/11-16:46:06.899175 [**] SMB Name Wildcard [**] 168.143.29.9:137 -> MY.NET.60.17:137
08/11-16:47:52.458244 [**] SMB Name Wildcard [**] 168.167.8.12:137 -> MY.NET.253.24:137
08/11-16:48:00.142286 [**] SMB Name Wildcard [**] 166.72.86.217:137 -> MY.NET.100.230:137
08/11-16:51:16.439948 [**] SMB Name Wildcard [**] 64.7.58.194:137 -> MY.NET.20.10:137
08/11-16:51:17.369342 [**] SMB Name Wildcard [**] 24.28.62.226:1975 -> MY.NET.70.121:137
08/11-16:52:02.576485 [**] SMB Name Wildcard [**] 209.150.98.231:137 -> MY.NET.130.91:137
08/11-16:52:02.576547 [**] SMB Name Wildcard [**] 209.150.98.231:137 -> MY.NET.130.91:137
08/11-16:56:11.373867 [**] SMB Name Wildcard [**] 168.143.29.9:137 -> MY.NET.60.17:137

If this traffic was internal to your network it would be less worrisome as you probably have several windows boxes sharing information, you should pay special attention to the external IP's that are connecting to more than one internal address. If there is no reason to have this traffic you might consider filtering this. For further info on the dangers of SMB check out

cve-1999-0225
cve-1999-0391
can-1999-0495
all found at cve.mitre.org

All in all you have a very large and active network with several alerts. It may be possible to minimize some of the false positives by taking a look at your afs and weeding out some of the gnutella and napster traffic. I would pay particular attention to the tiny fragments and verify the type of information on those hosts. You may want to consider installing some host based IDS products on some of your more mission critical machines.

Section 4 Methodology

After extracting the data from the SANS website I broke out the data into scans alerts and data where I used the cat command to merge all the files into their respective directories. Within the directories I chose to identify the most active hosts and or destinations. Using scripts found on Lenny Zeltser's practical, the data was broken into excel friendly format. Using excel I sorted the data by most active

alert or scan then by the host or attacker. The results of this sorting can be seen in the charts above. Based on the rankings that resulted from the charting of the snort data I chose to focus my analysis on the most active hosts and destinations, I tried to correlate the attacks and innocuous data with detects from www.google.com, www.giac.org and www.dogpile.com. All of the tools, commands and software packages that I used were fairly common such as Linux Mandrake, Ultraedit, Excel. Some of the commands that I used have already been identified in the above analysis but what I found most useful was using the diff command on data that I suspected was not malicious. Comparing data between to similarly active computers allowed me to visualize possible scenarios.

Last Updated: July 28th, 2017

Upcoming Training

SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201709,	Sep 11, 2017 - Oct 18, 2017	vLive
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
Baltimore Fall 2017 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Scottsdale SEC503	Scottsdale, AZ	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced