Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst (GCIA)

Practical Assignment
Version 3.2
Craig Baltes
Submitted: Oct 11, 2002

Part 1 – The State of Intrusion Detection

The Problem with NIDS.

Encryption

By incorporating encryption methods that are widely and freely available such as Openssl into Trojan programs the attacker can bypass most Network Intrusion Detection Systems (NIDS).

Back Orifice 2000 is a well-known example of a Trojan that can use strong encryption (3des). In the case of BO2k at least 2 popular NIDS, ISS Real Secure and Snort, have signatures to detect even encrypted BO2k traffic and engines that can brute force the encryption. However the methods that are used look for specific packet sizes, which can be changed, and the brute force decryption is VERY resource intensive.

Back Orifice is just one program that has encryption built in. There are wrapper programs such as Stunnel. “Stunnel can allow you to secure non-SSL aware daemons and protocols (like POP, IMAP, LDAP, etc) by having Stunnel provide the encryption, requiring no changes to the daemon's code.”. This opens up myriad possibilities for all Trojans old and new. I will be writing later about using chat as a control channel for Trojans, an Stunnel like wrapper would make detecting this virtually impossible.

Chat as a Control Channel

One development that I have seen personally is the use of IRC as a control channel. Corey Merchant of LURHQ corp. wrote a paper for security focus about a specific incident that I was involved in. The Trojan was installed using a well-publicized vulnerability in the Microsoft SQL server (CERT® Advisory CA-2002-22). The Trojan was comprised mostly of mIRC scripts that had custom triggers to perform many actions, from scanning a network for a particular service to erasing the Trojan’s host drive. In the case of the incident described by Corey the triggers were all pretty straight forward using words such as “attack” and “bombing” that allowed the analysts to easily identify malicious activity and write IDS signatures to match them:

```
# Stran is the op, our attacker. Here he sets the channel topic, which
# tells the zombies who to attack, and on what port(s).

:Stran!~stran@fbi.gov TOPIC #rubik :!0pana www.gotocasino.com 80

# A zombie on attbi.com's net reports that it is "bOmBiNg" the victim.

:xJ9XI58!~F107638I@rox-21043.atl.client2.attbi.com PRIVMSG #rubik
```

:5,1[#4bOmBiNg#5]#0-#3[#09 www.gotocasino.com #3]

The same zombie reports back a successful attack

:xJ9XI58!~F107638I@rox-21043.atl.client2.attbi.com PRIVMSG #rubik
:14Attacked host #15: #4 www.gotocasino.com #14port #15: #4 80

This goes on for many hosts, who continue to syn flood the victim.

It would have been just as easy however to obfuscate the trigger commands somewhat by using innocuous terms or nonsense words as the triggers. You could make the connection and chat session look like a normal conversation which would make it difficult for analysts to identify malicious traffic and nearly impossible to write NIDS signatures for. What I just described is VERY easy to do, as I said this Trojan was mostly a collection of mIRC scripts that I could modify in any text editor in about 5 minutes. Below is an example of what the previously described chat session could look like:

Stran is the op, our attacker. Here he sets the channel topic, which
tells the zombies who to attack, and on what port(s).

:Stran!~stran@fbi.gov TOPIC #rubik :check out www.gotocasino.com 80

A zombie on attbi.com's net reports that it is "bOmBiNg" the victim.

:xJ9XI58!~F107638I@rox-21043.atl.client2.attbi.com PRIVMSG #rubik
:5,1[#checking out#5]#0-#3[#09 www.gotocasino.com #3]

The same zombie reports back a successful attack

:xJ9XI58!~F107638I@rox-21043.atl.client2.attbi.com PRIVMSG #rubik
:14That site is GREAT! #15: #4 www.gotocasino.com #14port #15: #4 80

This goes on for many hosts, who continue to syn flood the victim.

As you can see it does not look nearly as malicious now as it did in its original form. The command that was executed was a DDOS command to attack port 80. This chat session would not trigger a NIDS and the outbound HTTP traffic would probably be permitted.

Raw IP Trojans

Working on this paper has brought me into contact with a program and a concept that has real possibility for defeating a NIDS easily. The “Q” Trojan by Mixter is a great example of a “Raw IP” Trojan. Rather than create a listener and use a standard client/server configuration the “Q” Trojan looks for raw IP packets that

match a specific parameter. The data of these packets will contain encrypted instructions that the “Q” daemon will perform. The following is a list of functions contained in the current version of “Q” (2.4) quoted from the README of the program:

> “RawIP Server Controller (qs) options:
>
> qs -C "command" server.com - Execute remote shell commands. The command can be any line that you could type on a shell, including system programs.
>
> qs -p port -S server.com - Make the server open an encrypted shell that you connect to with 'q', listening on the port specified with -p
>
> qs -p port -B "relay.com dport" server.com - Make the server open an encrypted bouncer on the specified port. If you connect to it, it will relay you to dport on relay.com.
>
> qs -U 99 server.com - All bouncer processes will be run under user id 99.
>
> qs -P /sbin/sash server.com - All encrypted shells will now use /sbin/sash as user shell instead of the default shell.
>
> qs -p port - This option specifies the destination port that a shell/bouncer server will listen on.
>
> qs -n - If you specify the -n option, the server will spawn a *NORMAL* shell or bouncer, running without any encryption at all, so that all clients can connect. These sessions are depreceated, you should use TRANSD mode instead.
>
> qs -i I or U or T - Chose a protocol, ICMP UDP or TCP for the RawIP activation packet. Only necessary if you need to use a specific protocol to bypass a firewall.
>
> qs -a authid - Same usage as q (both qs and q connections use CSA).
>
> qs -s source ip - Specify your source IP, else it will be random (only useful to bypass routing filters and firewalls).
>
> qs <command> targets - qs can message an unlimited amount of targets at once. You specify the target hosts/ip as the last arguments. Ex.: qs -C "shutdown -r now" fbi.gov cia.gov nsa.gov”

Here is an example of the "Q" Trojan at work on a real network. 10.0.0.2 is a Mandrake Linux version 8.2 system that has the "Q" daemon running (qd). 10.0.0.1

is a RedHat Linux version 7.3 system that I compiled the client/server package on and am using as the client. When designating a target or my commands I will use 1.2.3.4 as the IP address.

This is the output from the client:

```
[root@localhost Q2.4]# qs -C "telnet 1.2.3.4 31337" 10.0.0.2
[*] request: execute command 'telnet 1.2.3.4 31337'
[*] sending control packet to 10.0.0.2 (encrypted)
[root@localhost Q2.4]#
```

This is the output from a tcpdump:

```
[root@localhost Q2.4]# tcpdump -n -s0 -X -i eth0 not arp
tcpdump: listening on eth0

19:31:01.125611 216.127.166.0.1947 > 10.0.0.2.48017: S 0:148(148) ack 15487956 win 7678
0x0000   4500 00a8 fae3 0000 ca06 6cea d87f a600      E.........l.....
0x0010   0a00 0002 079b bb91 0000 0000 00ec 53d4      ..............S.
0x0020   0012 1dfe d35c 0000 4449 7a36 5a4e 705a      .....\..DIz6ZNpZ
0x0030   3374 396e 5763 6272 5437 5372 324a 7641      3t9nWcbrT7Sr2JvA
0x0040   414e 6c48 6d57 2f68 7757 4144 3143 3963      ANlHmW/hwWAD1C9c
0x0050   4451 3062 6672 3056 7367 5843 624f 6c4b      DQ0bfr0VsgXCbOlK
0x0060   662f 6e48 6e61 582f 302b 5255 522b 4877      f/nHnaX/0+RUR+Hw
0x0070   6431 7775 4d33 4c4e 686e 3067 2f49 4833      d1wuM3LNhn0g/IH3
0x0080   374f 676b 4e58 3971 754f 6662 4c74 3263      7OgkNX9quOfbLt2c
0x0090   6147 6178 7253 736b 7359 7034 642f 5639      aGaxrSsksYp4d/V9
0x00a0   4257 526e 6849 5875                          BWRnhIXu

19:31:01.169462 10.0.0.2.1043 > 1.2.3.4.31337: S 539291178:539291178(0) win 5840 <mss
1460,sackOK,timestamp 2015017 0,nop,wscale 0> (DF) [tos 0x10]
0x0000   4510 003c 5103 4000 4006 dba1 0a00 0002      E..<Q.@.@.......
0x0010   0102 0304 0413 7a69 2024 ee2a 0000 0000      ......zi.$.*....
0x0020   a002 16d0 d71c 0000 0204 05b4 0402 080a      ................
0x0030   001e bf29 0000 0000 0103 0300                ...).........

        4 packets received by filter
        0 packets dropped by kernel
```

This is another output from tcpdump when I ran the same "qs" command minutes later:

```
19:37:55.205588 143.63.86.0.30636 > 10.0.0.2.34916: S 13764962:13765110(148) ack 0 win
41869
0x0000   4500 00a8 d189 0000 f506 0485 8f3f 5600      E.............?V.
0x0010   0a00 0002 77ac 8864 00d2 0962 0000 0000      ....w..d...b....
0x0020   0012 a38d c9a5 0000 7056 796a 374b 4d73      ........pVyj7KMs
0x0030   7742 6566 3368 5647 3933 5869 4e70 7641      wBef3hVG93XiNpvA
0x0040   414e 6c48 6d57 2f68 7757 4144 3143 3963      ANlHmW/hwWAD1C9c
0x0050   4451 3062 6672 3056 7367 5843 624f 6c4b      DQ0bfr0VsgXCbOlK
0x0060   662f 6e48 6e61 582f 302b 5255 522b 4877      f/nHnaX/0+RUR+Hw
0x0070   6431 7775 4d33 4c4e 686e 3067 2f49 4833      d1wuM3LNhn0g/IH3
```

```
0x0080   374f 676b 4e58 3971 754f 6662 4c74 3263       7OgkNX9quOfbLt2c
0x0090   6147 6178 7253 736b 7359 7034 642f 5639       aGaxrSsksYp4d/V9
0x00a0   4257 526e 6849 5875                           BWRnhIXu

19:37:55.247334 10.0.0.2.1044 > 1.2.3.4.31337: S 969778169:969778169(0) win 5840 <mss
1460,sackOK,timestamp 2056423 0,nop,wscale 0> (DF) [tos 0x10]
0x0000   4510 003c c31e 4000 4006 6986 0a00 0002       E..<..@.@.i.....
0x0010   0102 0304 0414 7a69 39cd a3f9 0000 0000       ......zi9.......
0x0020   a002 16d0 65e5 0000 0204 05b4 0402 080a       ....e...........
0x0030   001f 60e7 0000 0000 0103 0300                 ..`.........

4 packets received by filter
        0 packets dropped by kernel
```

The first packet that you see in each capture is the RawIP control packet that contains the encrypted commands that I specified in my "qs" command. Without filling up this entire paper with packet captures of this Trojan in action suffice it to say that after extensive testing I have seen the client generate TCP, UDP, and ICMP RawIP control packets and have been able to find no common thread, or mistake that I could use to write a valid NIDS signature. In the captures above you see that the data portion of the RawIP control packet is the same in both captures but that is because the command is the same in both instances.

Authorized Access

The end-all, be-all of methods to bypass any security system is to have authorized access. A disgruntled employee who works in your personnel department can destroy or even worse publish all of the data on salaries. There have been a series of commercials by a popular security vendor on this subject lately, implying that by using their software you can eliminate this potential threat. Those of us who work in the security industry know however that this is probably our most real threat and the one that is most difficult to mitigate. As a security analyst the only times I have ever truly feared compromise is when my company or a customers company has lost an employee that had significant access. Where do you look? How do you know, this is not someone who will have to follow the normal methods of reconnaissance and compromise, this threat knows what he wants and how to get it with a minimum of exposure. We have the training, knowledge and resources to defend against most external threats, internal threats are like bogeymen. The only thing restraining the person with authorized access from using it for ill are the constraints of Civilization and Law and those can be cast aside at will and seem flimsy indeed compared to the good Firewall, NIDS and Security policy I can use to deal with other threats.

The Solution (Sort Of)

Host monitoring/Host based IDS

Running a host based IDS such as “Entercept” on critical servers is a good option. I

have personally used this software and found it to be very effective at stopping unauthorized access. Entercept actually intercepts all system calls and compares them to both a list of known malicious activity and a list of behavioral rules that dictate what a process is allowed to do. It can get in the way sometimes when you have a legitimate process being stopped, and it requires active administration to control it when that happens, but in all I would say that it is my personal choice as the best HIDS.

There are many other products that are worth mentioning as well. Tripwire, Dragon Squire, and Intruder alert. There are lots more products but to list them all and to delve deeper into HIDS is beyond the scope of this paper. I am mentioning them in relation to my original premise of problems with running NIDS and thinking that is enough.

Anti-Virus

One of the oldest and most taken for granted methods of prevention is having a well-designed and up to date antivirus architecture. This should contain a server side that scans well-known protocols like HTTP, FTP and SMTP as these protocols leave and enter your network, as well as host based antivirus that scans e-mail attachments and downloaded files. You should have a method of getting updates from your vendor automatically and also have a method of notification when new viruses with significant impact are found in the wild.

The threat of the “Q” Trojan I detailed earlier could have been avoided from the start if the method of loading the “qd” daemon was found. I believe this Trojan has been bundled with other packages such as a “hack a tack” as an attachment for an email which most antivirus clients would catch.

Real Time monitoring

So you have a good Firewall connecting you to the Internet, a well thought out NIDS/HIDS deployed on your internal network and DMZ, antivirus software is running on all your workstations and servers, and you have a clearly defined usage policy for your network resources. You are well-protected and should go home and get some deserved rest.

As you are resting a system administrator working late on your PDC has disabled the HIDS software because it was interfering with a software installation he is working on. He is installing a popular FTP software at the request of management so that your companies partners can upload documents directly to you, as you are using your PDC as your files server as well he figures it will make it easier for the users to get the files from there after they have been uploaded. He was unable to secure any funding to purchase supported software or to have a consultant who was familiar with the software install it, so he found a popular freeware package and is using the default installation. He is unaware that a remote exploit was released earlier that day for the software version he is installing. When he is done

installing he tests to see that he can indeed upload files and retrieve them on his workstation, and forgets completely to re-enable the HIDS. He is the authorized secondary administrator for your firewall and since he figures he is already at work and he can inform you tomorrow he can make the necessary changes to your firewall to make this server accessible from the internet at large. As he does not have the source address of all your companies partners he opens access to FTP from anywhere to the PDC. He then goes home, pleased that he has competed his task, and hoping that management will be satisfied.

Shortly after 1:00 am a large and noisy FTP scan originating in Korea sweeps by your internet address space, your new server dutifully responds that yes it is running FTP of a certain version. An exploit script is run against your server from Kazakhstan soon after, and it succeeds in compromising the FTP server. The Russian mafia hacker who is behind this is building a zombie army to…?

You get the picture, bad things just happened to you after a series of all too plausible events took place. All of the systems on your network provide valuable information, but its value is time sensitive. The information that your HIDS was disabled will do you no good tomorrow, nor will checking your snort logs and seeing the buffer overflow that compromised your server, or the firewall logs that show the ftp scan. By the time you are checking into this the next day, more systems could have been compromised on your network, more backdoors added, and more attacks committed using your network resources and causing you to possibly be liable for damages.

However if you had a third party or your own operations center to monitor all of the information provided by your security infrastructure this problem would never have happened. The HIDS outage would have been noticed and the appropriate contact would have been made with the sysadmin, who would have turned it back on when he was done. You would have then seen the FTP scan from your firewall logs and probably seen the exploit on both your NIDS and HIDS. The monitoring center would have called you, and the crisis could have been averted with no real damage done.

Part 2 - Network Detects

Detect 1

```
17:55:35.634488 255.255.255.255.31337 > 226.185.156.93.515: R [bad tcp cksum 8f8f!] 0:3(3)
ack 0 win 0 [RST cko] (ttl 14, id
0, len 43, bad cksum 9e27!)
0x0000   4500 002b 0000 0000 0e06 9e27 ffff ffff        E..+.......'....
0x0010   e2b9 9c5d 7a69 0203 0000 0000 0000 0000        ...]zi..........
0x0020   5014 0000 524f 0000 636b 6f00 0000             P...RO..cko...

18:08:14.624488 255.255.255.255.31337 > 226.185.19.65.515: R [bad tcp cksum 908d!] 0:3(3) ack
0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 2943!)
0x0000   4500 002b 0000 0000 0e06 2943 ffff ffff        E..+......)C....
0x0010   e2b9 1341 7a69 0203 0000 0000 0000 0000        ...Azi..........
0x0020   5014 0000 dd6a 0000 636b 6f00 0000              P....j..cko...

18:08:59.634488 255.255.255.255.31337 > 226.185.25.231.515: R [bad tcp cksum 908d!] 0:3(3)
ack 0 win 0 [RST cko] (ttl 14, id
0, len 43, bad cksum 229d!)
0x0000   4500 002b 0000 0000 0e06 229d ffff ffff        E..+......".....
0x0010   e2b9 19e7 7a69 0203 0000 0000 0000 0000        ....zi..........
0x0020   5014 0000 d6c4 0000 636b 6f00 0000             P.......cko...

18:15:32.624488 255.255.255.255.31337 > 226.185.59.113.515: R [bad tcp cksum 8f8f!] 0:3(3)
ack 0 win 0 [RST cko] (ttl 14, id
0, len 43, bad cksum ff13!)
0x0000   4500 002b 0000 0000 0e06 ff13 ffff ffff        E..+............
0x0010   e2b9 3b71 7a69 0203 0000 0000 0000 0000        ..;qzi..........
0x0020   5014 0000 b33b 0000 636b 6f00 0000              P....;..cko...

18:48:53.654488 255.255.255.255.31337 > 226.185.99.176.515: R [bad tcp cksum 8f8f!] 0:3(3)
ack 0 win 0 [RST cko] (ttl 14, id
0, len 43, bad cksum d6d4!)
0x0000   4500 002b 0000 0000 0e06 d6d4 ffff ffff        E..+............
0x0010   e2b9 63b0 7a69 0203 0000 0000 0000 0000        ..c.zi..........
0x0020   5014 0000 8afc 0000 636b 6f00 0000             P.......cko...

18:54:53.624488 255.255.255.255.31337 > 226.185.173.31.515: R [bad tcp cksum 8d90!] 0:3(3)
ack 0 win 0 [RST cko] (ttl 14, id
0, len 43, bad cksum 8c67!)
0x0000   4500 002b 0000 0000 0e06 8c67 ffff ffff        E..+.......g....
0x0010   e2b9 ad1f 7a69 0203 0000 0000 0000 0000        ....zi..........
0x0020   5014 0000 408f 0000 636b 6f00 0000              P...@...cko...

19:56:20.604488 255.255.255.255.31337 > 226.185.168.109.515: R [bad tcp cksum 8f8f!] 0:3(3)
ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 9217!)
0x0000   4500 002b 0000 0000 0e06 9217 ffff ffff        E..+............
0x0010   e2b9 a86d 7a69 0203 0000 0000 0000 0000        ...mzi..........
0x0020   5014 0000 463f 0000 636b 6f00 0000              P...F?..cko...
```

Source of Trace

http://www.incidents.org/logs/Raw

Detect was generated by

Snort IDS of indeterminate version.

Probability the source address was spoofed

The source address is spoofed. No legitimate IP traffic would come from a source address of 255.255.255.255. RFC 919 describes the recommended standards for use of 255.255.255.255 and it is as a local broadcast destination only, not as a source.

Description of attack

When I first ran this detect through snort I found that it did not trigger any alerts on these packets. This aroused my curiosity, since they are so obviously crafted packets with no legitimate use. They have a broadcast source address of 255.255.255.255, a source port of 31337 (heh heh), a ttl of 14, and a TCP RST flag set. I was curious as to what could possibly put a packet like this on a network, and why. I began to research online and found that other people had seen packets exactly like this and that snort USED to have a signature for this exact packet called "backdoor-Q". This signature is no longer listed on arachNIDS or snort.org. ArachNIDS does have a signature for "trojan-active-Q-tcp":

```
alert TCP 255.255.255.255/32 any -> $INTERNAL any (msg: "IDS203/trojan_trojan-active-Q-tcp";
dsize: >1; flags: A; classtype: system-success;  reference:arachnids,203;)
```

This is the only signature (apart from active-Q-udp and active-Q-icmp) that has a source address of 255.255.255.255. My packets were TCP RST packets and this signature only matched ACK packets. Still this being the only match for the source and having found other traffic similar to mine matching an older signature for the "Q" Trojan, I had enough information to warrant a closer look at this "Q" Trojan. ArachNIDS referenced a writer called "mixter" and I began to look for his site. I found Mixter's site and downloaded the Trojan from it. After looking at the README and various config files I have found that this is a VERY capable Trojan and is certainly able to duplicate my packets. Q can be compiled in numerous ways, one of which has the server listen for rawIP traffic matching a set type that was specified when it was compiled, when it sees this type of rawIP traffic it will perform an action, such as creating a shell on a specified port, or even opening an encrypted connection back to a specified host using SSL. Each compiled client/server "Q" Trojan set is unique and can only be used to communicate with each other.

It is my opinion that this traffic is raw IP traffic generated by a "Q" Trojan client which is scanning for servers. Whoever compiled this particular client/server set did a poor job of it. After looking at the configuration files and README for the "Q" Trojan it would be very easy to make this FAR less conspicuous.

Attack mechanism

The attack works by having a server listening for raw IP traffic on an infected host. When an infected host sees a "trigger" packet it will perform an action, such as opening a ssl connection back to a particular address or opening a shell on a specified port.

Correlation

I have seen evidence of traffic that could match this particular client/server set posted on many newsgroups.

I have also used proprietary sources at my place of business to discover similar traffic across at least 2 other networks.

Evidence of Active Targeting

The destination addresses seem to be random.

Severity

Criticality=2
I did not see any connections to the target systems other than the spoofed packets. I also did not see any outbound http or ssl attempt from the targeted hosts.

Lethality=5
If the spoofed packets did activate a raw IP Trojan, the infected system would be completely open to the attacker.

System countermeasures= 1
I have no information on the system countermeasures of any of the target systems.

Network countermeasures=1
I have no information on the network countermeasures of the target network.

Severity=5
This is probably higher than necessary but information would need to be gathered about the target network and hosts.

Defensive recommendation

Running a good firewall with anti spoofing rules would block this attack. Running up to date anti-virus software on your hosts would block the initial infection.

Incidents.org Questions

Oliver Viitamaki asked some very good questions about this detect. Here are his questions and my defense.

Q: Why do we have Bad TCP checksum? Is something being disguised?
A: Yes I believe that these are crafted packets. I was unable to replicate the "bad cksum" using the current "Q" Trojan but I think that we are seeing an older less capable version.

Note: I have since learned from Oliver that the bad cksum is due to the destination address being changed to obfuscate the true destination.

Q: Version, switch settings, network config? (For the source of trace)
A: I am unaware of the Version, switch, settings, network config as I received this detect from a online source. www.incidents.or/logs/RAW

Q: Why? (what is 255.255.255.255, in the context of legitimate)
A: Actually you quoted RFC 919 section 7 below, which is the exact RFC I would use to answer this question. The RFC provides that address for use as a DESTINATION when a host wishes to SEND a packet of data to all of it's neighbors. The SOURCE address of a packet from a system that did not have an IP address assigned would be 0.0.0.0 or "unspecified". The response to a destination 255.255.255.255 would be from the IP address of the host interface that received the broadcast. It is possible to craft packets with a source of 255.255.255.255 but this is not compliant with RFC 919 as quoted below.

Q: Are there any simpler choices, than using a Firewall for this purpose? (This question is related to my defensive recommendations)
A: Firewalls that have anti-spoofing capabilities (As most do today) are really the best choice to block this traffic.

Q: For the sake of the following question lets assume that the IDS is in the DMZ along with a number of other machines, one of which is compromised, and is actually sending out the traffic as a local Broadcast to the 226.185.0.0 Network. Would this change any of the conclusions, observations, severity, recommendations?
A: The destination address is NOT a local broadcast. It is the source which is a "limited broadcast address" as defined by TCP/IP illustrated Volume one p. 171. The scenario you suggest would definitely change my conclusions as the

"Q" Trojan does not use broadcast addresses as it's destination. You must provide specific source address for the client to send to the "Q" server.

Multiple choice test question

```
17:55:35.634488 255.255.255.255.31337 > 226.185.156.93.515: R [bad tcp cksum 8f8f!] 0:3(3)
ack 0 win 0 [RST cko] (ttl 14, id
0, len 43, bad cksum 9e27!)
0x0000   4500 002b 0000 0000 0e06 9e27 ffff ffff        E..+.......'....
0x0010   e2b9 9c5d 7a69 0203 0000 0000 0000 0000        ...]zi..........
0x0020   5014 0000 524f 0000 636b 6f00 0000              P...RO..cko...
18:08:14.624488 255.255.255.255.31337 > 226.185.19.65.515: R [bad tcp cksum 908d!] 0:3(3) ack
0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 2943!)
0x0000   4500 002b 0000 0000 0e06 2943 ffff ffff        E..+......)C....
0x0010   e2b9 1341 7a69 0203 0000 0000 0000 0000        ...Azi..........
0x0020   5014 0000 dd6a 0000 636b 6f00 0000              P....j..cko...
18:08:59.634488 255.255.255.255.31337 > 226.185.25.231.515: R [bad tcp cksum 908d!] 0:3(3)
ack 0 win 0 [RST cko] (ttl 14, id
0, len 43, bad cksum 229d!)
0x0000   4500 002b 0000 0000 0e06 229d ffff ffff        E..+......".....
0x0010   e2b9 19e7 7a69 0203 0000 0000 0000 0000        ....zi..........
0x0020   5014 0000 d6c4 0000 636b 6f00 0000              P.......cko...
18:15:32.624488 255.255.255.255.31337 > 226.185.59.113.515: R [bad tcp cksum 8f8f!] 0:3(3)
ack 0 win 0 [RST cko] (ttl 14, id
0, len 43, bad cksum ff13!)
0x0000   4500 002b 0000 0000 0e06 ff13 ffff ffff        E..+............
0x0010   e2b9 3b71 7a69 0203 0000 0000 0000 0000        ..;qzi..........
0x0020   5014 0000 b33b 0000 636b 6f00 0000              P....;..cko...
18:48:53.654488 255.255.255.255.31337 > 226.185.99.176.515: R [bad tcp cksum 8f8f!] 0:3(3)
ack 0 win 0 [RST cko] (ttl 14, id
0, len 43, bad cksum d6d4!)
0x0000   4500 002b 0000 0000 0e06 d6d4 ffff ffff        E..+............
0x0010   e2b9 63b0 7a69 0203 0000 0000 0000 0000        ..c.zi..........
0x0020   5014 0000 8afc 0000 636b 6f00 0000              P.......cko...
18:54:53.624488 255.255.255.255.31337 > 226.185.173.31.515: R [bad tcp cksum 8d90!] 0:3(3)
ack 0 win 0 [RST cko] (ttl 14, id
0, len 43, bad cksum 8c67!)
0x0000   4500 002b 0000 0000 0e06 8c67 ffff ffff        E..+.......g....
0x0010   e2b9 ad1f 7a69 0203 0000 0000 0000 0000        ....zi..........
0x0020   5014 0000 408f 0000 636b 6f00 0000              P...@...cko...
19:56:20.604488 255.255.255.255.31337 > 226.185.168.109.515: R [bad tcp cksum 8f8f!] 0:3(3)
ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 9217!)
0x0000   4500 002b 0000 0000 0e06 9217 ffff ffff        E..+............
0x0010   e2b9 a86d 7a69 0203 0000 0000 0000 0000        ...mzi..........
0x0020   5014 0000 463f 0000 636b 6f00 0000              P...F?..cko...
```

Which of the following is MOST LIKELY true about the above trace?

a) This is a Print server Denial of Service attack
b) This is a scan for vulnerable print servers
c) This shows a Back Orifice established connection

d)These are crafted packets

Answer: d

Detect 2

Jul 21 00:47:55 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2528 to THIS.NET.121.67 on unserved port 1433
Jul 21 00:47:55 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2530 to THIS.NET.8.11 on unserved port 1433
Jul 21 00:47:55 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2529 to THIS.NET.8.10 on unserved port 1433
Jul 21 00:47:55 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2531 to THIS.NET.8.12 on unserved port 1433
Jul 21 00:47:55 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2532 to THIS.NET.8.13 on unserved port 1433
Jul 21 00:47:55 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2533 to THIS.NET.8.14 on unserved port 1433
Jul 21 00:47:55 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2534 to THIS.NET.8.15 on unserved port 1433
Jul 21 00:47:55 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2535 to THIS.NET.8.16 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2528 to THIS.NET.121.67 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2537 to THIS.NET.8.17 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2538 to THIS.NET.8.19 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2536 to THIS.NET.8.18 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2530 to THIS.NET.8.11 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2541 to THIS.NET.60.241 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2529 to THIS.NET.8.10 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2539 to THIS.NET.8.20 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2540 to THIS.NET.8.21 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2546 to THIS.NET.121.85 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2547 to THIS.NET.121.86 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2531 to THIS.NET.8.12 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2532 to THIS.NET.8.13 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2549 to THIS.NET.121.88 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2550 to

THIS.NET.121.90 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2533 to THIS.NET.8.14 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2534 to THIS.NET.8.15 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2552 to THIS.NET.50.17 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2553 to THIS.NET.121.92 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2535 to THIS.NET.8.16 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2528 to THIS.NET.121.67 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2538 to THIS.NET.8.19 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2530 to THIS.NET.8.11 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2537 to THIS.NET.8.17 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2541 to THIS.NET.60.241 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2539 to THIS.NET.8.20 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2536 to THIS.NET.8.18 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2529 to THIS.NET.8.10 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2546 to THIS.NET.121.85 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2547 to THIS.NET.121.86 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2540 to THIS.NET.8.21 on unserved port 1433
Jul 21 00:47:56 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2531 to THIS.NET.8.12 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2532 to THIS.NET.8.13 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2549 to THIS.NET.121.88 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2550 to THIS.NET.121.90 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2533 to THIS.NET.8.14 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2534 to THIS.NET.8.15 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2552 to THIS.NET.50.17 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2535 to THIS.NET.8.16 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2553 to THIS.NET.121.92 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2541 to THIS.NET.60.241 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2536 to

THIS.NET.8.18 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2538 to THIS.NET.8.19 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2537 to THIS.NET.8.17 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2539 to THIS.NET.8.20 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2540 to THIS.NET.8.21 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2547 to THIS.NET.121.86 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2546 to THIS.NET.121.85 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2549 to THIS.NET.121.88 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2550 to THIS.NET.121.90 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2552 to THIS.NET.50.17 on unserved port 1433
Jul 21 00:47:57 gateway.THIS.NET.com unix: securityalert: tcp if=hme0 from 211.176.61.88:2553 to THIS.NET.121.92 on unserved port 1433

Source of trace

This trace was taken from a database at my place of business. I work for a Managed Security Services Provider that catalogs an correlates events from multiple devices such as Firewalls and NIDS. I have obfuscated the customer specific information and I have obtained permission to use the data.

Detect was generated by

Network Assoiciates Gauntlet firewall version 5.5.

Probability the source address was spoofed

The source address is not likely to be spoofed. The attacker needs to see responses in order for this attack to succeed. There is always the possibility that this is spoofed and the attacker is looking for responses on the local network or somewhere on the route back, I feel this is unlikely however do to the straightforward nature of the scan. If an attacker were to have the time and expertise to intercept these packets on a local network or on a router in the path, they would more likely use a more stealthy scan. This is valid host that can be pinged and tracerouted to.

IP Address : 211.176.61.0-211.176.61.255
Network Name : HANANET-NGENE
Connect ISP Name : HANANET
Connect Date : 20000628
Registration Date : 20000708

[Organization Information]
Orgnization ID : ORG126663
Org Name : HANARO Telecom
State : SEOUL
Address : 1445-3 Seocho-Dong Seocho-Ku
Zip Code : 137-728

DShield Profile:
Country: KR
Contact E-mail: abuse_AT_hananet.net (bounced)
Total Records against IP: 380
Number of targets: 200
Date Range: 2002-07-20 to 2002-07-20
Ports Attacked (up to 10):

Port	Attacks
1433	1

Description of attack

This attack is probably the "Spida" worm described in CERT® Incident Note IN-2002-04:
"Reports received by the CERT/CC indicate that the Spida worm scans for systems listening on port 1433/tcp. Once connected, it attempts to use the
xp_cmdshell utility to enable and set a password for the guest user.
If successful, the worm then

1.assigns the guest user to the local Administrator and Domain Admins groups
2.copies itself to the victim system
3.disables the guest account
4.sets the sa password to the same password as the guest account
5.executes the copy on the victim system

Once the local copy is executing on the victim system, the worm begins scanning for other systems to infect. It also attempts to send a copy of the local password (SAM) database, network configuration information, and other SQL server configuration information to a fixed email address (ixtld@postone.com) via email."

or the "Kaiten" malicius code described in CERT® Incident Note IN-2001-13:

"'Kaiten' made its initial appearance in August 2001 and is based on the "Knight" distributed attack tool mentioned in CA-2001-20 Continuing Threats to Home Users.

In reports received by the CERT/CC, installation of "Kaiten" was preceded by scans for hosts listening on 1433/tcp (MS-SQL). The infection process leverages sa accounts with null passwords to gain access to vulnerable systems. It then uses the xp_cmdshell stored procedure to initiate an FTP session from the victim system to a remote site. A copy of "Kaiten" is then downloaded and executed on the victim system.

Additional information on the null default sa password in Microsoft SQL Server, MSDE, and MMS is available in VU#635463.

Once the "Kaiten" code has begun execution on the victim system, it connects to an IRC server (on port 6667/tcp or 6669/tcp, according to reports received by the CERT/CC) to await further commands from the attacker. The attacker can then remotely issue commands to multiple compromised systems simultaneously, allowing compromised hosts to be used as DDoS agents, port scanners, etc. The attacker can also remotely reconfigure "Kaiten" via IRC to modify certain settings, including the IRC servers and channels it connects to."

I do not see a way of classifying it as one or the other. Both attacks contain scan elements and the only information I have is scans on port 1433 from the attacking host. However I do have evidence that a "Kaiten" like tool was used in the recent past to compromise a home user that connects into this network via VPN. The home user was running ms-sql with no password and the system was attempting connections to an IRC server in Kazakhstan. I can speculate that this put the attacked network into view as a possible hunting ground for more vulnerable systems.

Attack mechanism

Both these attacks use weak or empty "sa" passwords on ms-sql servers to compromise their targets.

Correlations

This attacker has been observed before on a different network:
time=25Jun2002 7:19:52 action=drop orig=THAT.NET.16.178 i/f_dir=inbound i/f_name=eth-s4p1c0 proto=tcp src=211.176.61.88 dst=THAT.NET.226.31 service=1433 s_port=2350
time=25Jun2002 7:19:52 action=drop orig=THAT.NET.16.178 i/f_dir=inbound i/f_name=eth-s4p1c0 proto=tcp src=211.176.61.88 dst=THAT.NET.226.30 service=1433 s_port=2349
time=25Jun2002 7:19:52 action=drop orig=THAT.NET.16.178 i/f_dir=inbound i/f_name=eth-s4p1c0 proto=tcp src=211.176.61.88 dst=THAT.NET.226.28 service=1433 s_port=2347
time=25Jun2002 7:19:52 action=drop orig=THAT.NET.16.178 i/f_dir=inbound i/f_name=eth-s4p1c0 proto=tcp src=211.176.61.88 dst=THAT.NET.226.22 service=1433 s_port=2341
time=25Jun2002 7:19:52 action=drop orig=THAT.NET.16.178 i/f_dir=inbound i/f_name=eth-s4p1c0 proto=tcp src=211.176.61.88 dst=THAT.NET.226.26 service=1433 s_port=2346
time=25Jun2002 7:19:52 action=drop orig=THAT.NET.16.178 i/f_dir=inbound i/f_name=eth-s4p1c0 proto=tcp src=211.176.61.88 dst=THAT.NET.226.23 service=1433 s_port=2342

This attacker has also been reported to Dshield:

Country:	KR
Contact E-mail:	abuse_AT_hananet.net (bounced)
Total Records against IP:	380
Number of targets:	200
Date Range:	2002-07-20 to 2002-07-20
Ports Attacked (up to 10):	

Port	Attacks

1433 1

Evidence of active targeting

There is no evidence of active targeting from this attacker, however I did mention that a host on the target network had been compromised before using a similar attack before and this may have bearing on why they were targeted.

Severity

Criticality = 5
The target network contains many different types of systems including web servers, firewalls, and DNS servers
Lethality = 5
If this attack was successful it would result in a total system compromise.
System Countermeasures = 1
The security of the target system is unknown.
Network Countermeasures = 5
The network is protected by a proxy based firewall with a recently audited configuration. They also have a SNORT IDS on this network and both of these are monitored 24/7.
Severity = 4

Defensive recommendation

I would recommend that ms-sql port 1433 be blocked for inbound access (which it is)
I would also recommend that all users of ms-sql have their systems audited for evidence of current compromise and/or vulnerabilities pursuant to the recommendations in CERT® Incident Note IN-2001-13 and CERT® Incident Note IN-2002-04.

Mulitple choice test question

Which of the following is a step you can take to protect against the "Spida" worm?
a. Block inbound access on TCP port 3306
b. Block outbound email to "spida@hotmail.com"
c. Block outbound access on TCP port 6666
d. Block outbound email to "ixtld@postone.com"
answer: d

Detect 3

Jul 5 15:03:35 THAT.NET.org gfw: [ID 702911 kern.info] securityalert: tcp if=qfe1 from

24.132.105.2:1914 to THIS.NET.152.228 on unserved port 12345
Jul 5 15:03:34 THAT.NET.org gfw: [ID 702911 kern.info] securityalert: tcp if=qfe1 from 24.132.105.2:1913 to THIS.NET.152.228 on unserved port 27374
Jul 5 15:03:34 THAT.NET.org gfw: [ID 702911 kern.info] securityalert: tcp if=qfe1 from 24.132.105.2:1908 to THIS.NET.152.227 on unserved port 12345
Jul 5 15:03:34 THAT.NET.org gfw: [ID 702911 kern.info] securityalert: tcp if=qfe1 from 24.132.105.2:1906 to THIS.NET.152.227 on unserved port 27374
Jul 5 15:03:33 THAT.NET.org gfw: [ID 702911 kern.info] securityalert: tcp if=qfe1 from 24.132.105.2:1904 to THIS.NET.152.226 on unserved port 27374
Jul 5 15:03:34 THAT.NET.org gfw: [ID 702911 kern.info] securityalert: tcp if=qfe1 from 24.132.105.2:1905 to THIS.NET.152.226 on unserved port 12345
Jul 5 15:03:32 THAT.NET.org gfw: [ID 702911 kern.info] securityalert: tcp if=qfe1 from 24.132.105.2:1897 to THAT.NET.51.98 on unserved port 12345
Jul 5 15:03:32 THAT.NET.org gfw: [ID 702911 kern.info] securityalert: tcp if=qfe1 from 24.132.105.2:1896 to THAT.NET.51.98 on unserved port 27374

Source of trace

This trace was taken from a database at my place of business. I work for a Managed Security Services Provider that catalogs an correlates events from multiple devices such as Firewalls and NIDS. I have obfuscated the customer specific information and I have obtained permission to use the data.

Detect was generated by

Network Associates Gauntlet firewall version 6.0

Probability the source address was spoofed

The address was probably not spoofed. This attack requires a response in order to succeed. There are methods that can be employed to spoof addresses and still see responses, such as hacking a router and adding route advertisements, then removing them when you are done, http://www.incidents.org/archives/intrusions/msg01757.html.

They also could have access to another host on the local network and be looking for responses there.

These methods are far more complicated and using "Ockham's Razor" we find that it is more likely that the address is not spoofed.

```
inetnum:      24.132.104.0 - 24.132.105.255
              netname:      UPC-A2000-AMSTERDAM3
              descr:        UPC/A2000/ Kabeltelevisie Amsterdam
              descr:        regio Amsterdam 3
              country:      NL
              admin-c:      RIHU1-RIPE
              tech-c:       RIHU1-RIPE
              tech-c:       FA1780-RIPE
              status:       ASSIGNED PA
```

notify: hostmaster@a2000.com
mnt-by: A2000-KTA-MNT
changed: R.Huisman@a2000.com 20000105
changed: R.Huisman@a2000.com 20001113
source: RIPE

Description of attack

This a scan for systems that have been previously infected with either the SubSeven or NetBus Trojan's. CERT® Incident Note IN-2001-07:

"The CERT/CC has received an increasing number of reports regarding the compromise of home user machines running Microsoft Windows. Most of these reports surround the intruder tool SubSeven. SubSeven is often used as a Trojan horse, which allows an intruder to deliver and execute any custom payload and run arbitrary commands on the affected machine. This control includes the ability to read, modify, and delete confidential information. Additionally, the intruder may use the affected computer as a launching point for additional attacks (namely, denial of service).

While we believe that this level of intruder activity is not unusual, additional concern may be warranted in light of a new emerging class of "malware" such as W32/Leaves. W32/Leaves appears to be representative of a class of self-replicating, malicious code that automatically scans for hosts with these toolkits installed and leverages backdoors (i.e., SubSeven) for further malicious activity. An existing backdoor installed on a host by one intruder can now be used by another without any prior communication or intention for collaboration between intruders."

This attack is probably not W32/Leaves as it does not look just for SubSeven, but NetBus as well. It is very likely that this is one of "emerging class of 'malware'" that CERT describes. Why go through all of the trouble of finding vulnerable hosts, compromising them, etc. when you can just use one of the legion of zombies that have been abandoned or forgotten.

The W32.chainsaw worm also exhibits similar behavior. It scans networks looking for Netbus or SubSeven as well as open NETBIOS shares.
http://securityresponse.symantec.com/avcenter/venc/data/w32.chainsaw.worm.html

Attack mechanism

The attack works by scanning for hosts previously compromised by other trojans and uses them for it's own purposes. It may be a variant of the W32/Leaves toolkit or W32.chainsaw worm.

Correlations

The following logs were generated by a Checkpoint NG firewall:
time=26Jun2002 9:58:57 action=drop orig=THAT.NET.183.10 i/f_dir=inbound i/f_name=eth0 has_accounting=0 product=VPN-1 & FireWall-1 src=24.132.105.2 s_port=4050 dst=THAT.NET.183.10 service=27374 proto=tcp

time=26Jun2002 9:58:57 action=drop orig=THAT.NET.183.10 i/f_dir=inbound i/f_name=eth0 has_accounting=0 product=VPN-1 & FireWall-1 src=24.132.105.2 s_port=4051 dst=THAT.NET.183.10 service=12345 proto=tcp

DShield Profile:

Country:	NL
Contact E-mail:	abuse@a2000.nl
Total Records against IP:	744
Number of targets:	119
Date Range:	2002-07-06 to 2002-07-06

Evidence of active targeting

I have no evidence of active targeting. The large number of scanned hosts and correlating evidence suggests that this is a general scan.

Severity

Criticality = 5
The target of the scan includes the firewall.
Lethality = 5
Were this scan to find a compromised host then the attacker would have full access the the target system.
System Countermeasures = Unknown
The scan covered a range of systems.
Network countermeasures = 5
They have a proxy based firewall with an audited ruleset that is monitored 24/7.
Severity = 4

Defensive recommendation

The network has a strong firewall which blocked the attack. I would recommend a Network IDS such as SNORT be utilized as well as the firewall.

Multiple choice test question

Which method of propagation that the W32.chainsaw uses would be most effective on a Local Area Network that has non-routable addresses and is behind a firewall?

a. SubSeven scanning
b. NetBus scanning
c. NETBIOS share scanning

Answer: C

Part 3 Analyze This

Executive Summary

During the 5 days of logs that I analyzed for MY.NET 1,538,805 events occurred. The most serious problem I found was the amount of false positives. In the outbound UDP scans section of my analysis alone there were 910,000 events that were detected as “scans” that I analyzed as being online gaming and file sharing. This is a common theme throughout my analysis. Some major care and feeding needs to be done to increase the effectiveness of the MY.NET Snort NIDS infrastructure. As it stands the amount of false positives that are being generated are a hindrance to your information security analysts. Out of the 6 alerts that generated more than 2,000 events the top 4 were analyzed as “noise”. Out of the 4 scan events that generated more than 10,000 events the first section alone as I described above contains 910,000 “noise” events.
This does not mean that interesting events did not occur during the 5 day period, it just means that you have to look harder and risk missing some of the more meaningful events. I did analyze 6 events related to possible Trojan activity, 5 of these events are serious to varying degrees. There is a possible SubSeven infection on MY.NET as well as a MYPARTY infection, there is also some suspicious use of Internet Relay Chat (IRC). The Out Of Spec files indicate that the increase in use of the Explicit Congestion Notification Standard (ECN) is increasing the false positives in what is normally a very low occurrence event. All in all I would say that MY.NET has fair visibility with its current Snort NIDS infrastructure that could be improved greatly with some tuning.

List of files

alert.020917
alert.020918
alert.020919
alert.020920
alert.020921

scans.020917
scans.020918
scans.020919
scans.020920
scans.020921

oos_Jun.12.2002
oos_Jun.14.2002
oos_Jun.15.2002
oos_Jun.19.2002
oos_Jun.20.2002
oos_Jun.21.2002
oos_Jun.3.2002
oos_Jun.5.2002

Alerts Reported More Than 2000 Times

Occurrences	Alert	Severity
76147	spp http decode: IIS Unicode attack detected	Noise
42273	SUNRPC highport access!	Noise
11863	Watchlist 000220 IL-ISDNNET-990517	Noise
7688	spp http decode: CGI Null Byte attack detected	Noise
4910	Incomplete Packet Fragments Discarded	High
2458	SMB Name Wildcard	High

Frequent alert analysis

IIS Unicode attack detected

Severity: Noise
Occurrences: 76,147
Snort signature ID: http_decode

Summary: Code Red, Nimda, and the sadmind worms all exploit the well-known flaw in IIS described in CERT Vulnerability Note VU#111677. This alert is well-known for its false positives and were this a real infection I would expect to see a large amount of correlating alerts for “cmd.exe” and “root.exe” use by the same hosts that caused the Unicode alerts. On a network where such a large amount of web browsing occurs I would recommend using the “-unicode” option on the http_decode line of your Snort configuration file. Some may argue that this will reduce your visibility, but I think that wasting time on 76,147 events is reducing your visibility even worse. There already exist Snort signatures such as 1002, 1256, and 989 which look at packet content and identify the major worms and scripts that utilize the Unicode vulnerability. Good signature management and ability to write your own snort signatures can arguably replace the need for the Unicode function in snort.

Correlation: Todd Beardsley analyzed similar traffic in his GCIA paper of 31 May 2002. His analysis indicated that the 26,048 events of this type were a serious infection and he gave detailed and accurate recommendations on how to combat it. He did not however correlate non http_decode signatures into his analysis.

Recommendations: As I stated earlier I would disable the Unicode function of the http_decode preprocessor and rely on snort signatures written to detect content that matches “in the wild” malicious programs. Your ability to write good snort

signatures and the rapid response of the snort community to new outbreaks should reduce the risk of this move to an acceptable level.

SUNRPC highport access!

Severity: Noise
Occurrences: 42273
Snort signature ID: None

Summary: This signature seems to be matching any established packets with a destination port of 32771. The ARACHNIDS database contains several signatures of this nature and all are based on established packets from any source port to ports used by RPC. Snort signatures of this type are noise makers on a network where there are large amounts of Internet traffic. Signatures of this nature are notorious for generating false positives. Most of the NIDS community has moved away from signatures based on packet state and port number, as an example I cannot find a current signature in the snort.org database that is similar to this one. The current signatures for SUNRPC traffic are much more specific like Snort signature ID 1278 (RPC rstatd query), this signature matches on specific content as well as port. As a further evidence of this alert being just noise I would point out that of the 42,273 occurrences 37,406 were from this source address "152.163.134.164". Connecting a web browser to this address reveals this message:

ICY 404 Resource Not Found icy-notice1:SHOUTcast Distributed Network Audio Server/posix v1.8.1 icy-notice2:The resource requested was not found

So this was someone listening to internet radio not someone exploiting a SUNRPC vulnerability. The source network block is also owned by AOL:

OrgName: America Online
OrgID: AOL

NetRange: 152.163.0.0 - 152.163.255.255
CIDR: 152.163.0.0/16
NetName: AOL-BNET
NetHandle: NET-152-163-0-0-1
Parent: NET-152-0-0-0-0
NetType: Direct Assignment
NameServer: DNS-01.NS.AOL.COM
NameServer: DNS-02.NS.AOL.COM
Comment:
RegDate: 1992-04-01
Updated: 1999-12-02

TechHandle: AOL-NOC-ARIN
TechName: America Online, Inc.
TechPhone: +1-703-265-4670

TechEmail: domains@aol.net

ARIN Whois database, last updated 2002-10-04 19:05
Enter ? for additional hints on searching ARIN's Whois database.

Correlation: Michael Holstein in his GCIA paper of 30 June 2002 said that "This alert indicates a successful connection to the RPC service on a Solaris host". However this alerts does NOT indicate anything of the sort. This signature has no method for detecting the OS of the target and does not indicate a successful connection to any service. It merely indicates that an established packet has been seen with a destination port that matches this signature.

Recommendations: I would recommend that your Snort signatures and engine be updated to eliminate signatures like this one. The snort community is constantly growing and gaining experience, and you need to keep abreast of developments and changes in methods to more effectively manage your NIDS infrastructure.

Watchlist 000220 IL-ISDNNET-990517

Severity: Noise
Occurrences: 11863
Snort signature ID: None

Watchlist 000220 IL-ISDNNET-990517 [**] PT712022.bezeqint.net:80 -> MY.NET.182.91:1165
Watchlist 000220 IL-ISDNNET-990517 [**] w4.incredimail.com:80 -> MY.NET.139.10:1914
Watchlist 000220 IL-ISDNNET-990517 [**] thewho.schoolsucks.com:80 -> MY.NET.138.43:1259
Watchlist 000220 IL-ISDNNET-990517 [**] cablep-179-98-55.cablep.bezeqint.net:4384 -> MY.NET.163.107:1214
Watchlist 000220 IL-ISDNNET-990517 [**] PT712017.bezeqint.net:80 -> MY.NET.182.91:1633
Watchlist 000220 IL-ISDNNET-990517 [**] bzq-179-35-98.dcenter.bezeqint.net:80 -> MY.NET.83.247:1151
Watchlist 000220 IL-ISDNNET-990517 [**] bzq-179-35-121.dcenter.bezeqint.net:80 -> MY.NET.153.185:1079
Watchlist 000220 IL-ISDNNET-990517 [**] www.incredibarvuz1.com:80 -> MY.NET.150.223:1103
Watchlist 000220 IL-ISDNNET-990517 [**] cablep-179-99-36.cablep.bezeqint.net:2842 -> MY.NET.84.147:1214

Summary: This signature seems to match a "Watchlist" of Israeli networks. I question the need for such a signature at all. Most of the traffic I observed matching this signature is from MY.NET addresses web browsing and file sharing with addresses in Israel. If you are concerned with activity from this network then block it, or if you want statistics for network usage to this network then log it on your firewall or router, but don't make a snort signature for it. Without knowing the reasons for deploying this signature I cannot effectively analyze it.

Correlations: Kevin Timm in his GCIA practical of 31 May 2002 eliminated these events from his analysis. However I think that even a short analysis such as mine that questions the validity of the alert helps to further the effort to eliminate signatures such as these.

Recommendations: I recommend that this signature be removed. If this is used for data gathering purposes then a method should be implemented on another network device, such as a firewall, to gather the relevant data. To continue to deploy a signature of this nature on a NIDS infrastructure already struggling with huge amounts of false positives is not recommended.

CGI Null Byte attack detected

Severity: Noise
Occurrences: 7688
Snort signature ID: None

Summary: I actually work with Joe Stewart, the person who wrote this plugin and we have discussed this at length. He says:

"Basically, if the http decoding routine finds a %00 in an http request, it will alert with this message. Sometimes you may see false positives with sites that use cookies with urlencoded binary data, or if you're scanning port 443 and picking up SSLencrypted traffic"

Here is another quote from the snort FAQ:

"Q: I am getting too many "IIS Unicode attack detected" and/or "CGI Null Byte attack detected" false positives. How can I turn this detection off?
A: These messages are produced by the http_decode preprocessor. If you wish to turn these checks off, add -unicode or -cginull to your http_decode preprocessor line respectively.

preprocessor http_decode: 80 8080 -unicode -cginull

Your own internal users normal surfing can trigger these alerts in the preprocessor. Netscape in particular has been known to trigger them. Instead of disabling them,try a BPF filter to ignore your outbound http traffic such as:
snort -d -A fast -c snort.conf not (src net xxx.xxx and dst port 80)
This has worked very well for us over a period of 5-6 months and Snort is still very able to decode actual and dangerous cgi null and unicode attacks on our public web servers."

So there we have it straight from the horse's mouth. This signature is known to cause false positives with a Netscape cookie. The vast majority of these alerts were from internal to external and even the few that were not are suspect. Without the actual content of the packet there is no way to tell if these were actual attacks or false positives.

Correlations: Bradley Urwiller in his GCIA practical of 24 May 2002 comes to the exact same conclusions as I have. He even quotes the snort FAQ. Other than coming up with the correlations of Joe Stewart's posts to the snort mailing lists his analysis was perfect.

Recommendations: I would recommend that the steps outlined in the snort FAQ are implemented.

Incomplete Packet Fragments Discarded

Severity: High
Occurrences: 4910
Snort signature ID: None

This alert is generated by the defrag preprocessor. There have been numerous complaints about the defrag preprocessor causing false positives and even crashing snort. They have replaced defrag with the frag2 preprocessor and Marty Roesch himself recommends switching to it. When I began this analysis I had assumed that this traffic was noise but I now think that a very close look needs to be given to at least one of the targets matching this alert. MY.NET.153.197 was the target of 3616 of the 4910 alerts of this type. All of the alerts targeted at MY.NET.153.197 originated from 61.155.103.178. Dshield.org does not show the source as a known attacker. I think that the packets may be ICMP fragments with a TCP header hidden in them which some older unpatched OS's will reassemble. There is a very good description of this type of exploit that was posted to bugtraq in 1997 http://www.insecure.org/sploits/NT.no_first_fragment.IP_frag.attack.htm. Without seeing the content of the packets (or fragments) I am unable to verify this information; however I cannot think of a legitimate reason for so many fragments unless there is a serious network problem.

Correlation: Edward Peck in his GCIA practical of 25 November 2001 analyzed the events briefly stating that they were probably a network error but may be an attempt at reconnaissance.

Recommendations: I would recommend that you block the source address on your firewall or router and thoroughly investigate the target host. Even if you do not find anything, if you determine that the system is vulnerable and or has been vulnerable to publicly available exploits then it should be rebuilt.

SMB Name Wildcard

Severity: High
Occurrences: 2458
Snort signature ID:

Summary: This event is probably noise if it originates from the internal network as some of it does. However 1169 of these events originate from 80.11.207.237. I believe that these are legitimate attempts to find systems that are vulnerable to the recent Windows SMB vulnerabilities that are detailed in CERT vulnerability notes VU#250635, VU#311619, and VU#342243.

Correlations: Todd Beardsley in his GCIA practical of 31 May 2002 analyzed events of this type as noise and suggested that it was normal NetBIOS nameserver traffic. I think that in May that this analysis was correct but in the ever changing landscape of network security, these events have gone from noise to evil. Dshield.org lists 80.11.207.237 755 times, all of which are scans to port 1433, probably looking for systems that are vulnerable to another recent Microsoft vulnerability that exists in

Microsoft SQL server.

Recommendations: All NetBIOS traffic should be blocked at your border router, both inbound and outbound. Your network administrators should have guidelines that your students are aware of that detail how to set up secure systems. The network administrators should also have a robust and comprehensive patch management scheme in place so that they can roll out patches quickly and easily.

Scans Reported More Than 10000 Times

Occurrences	Alert
972972	UDP Scan (Outbound)
63685	UDP Scan (Inbound)
17802	SYN Scan (Outbound)
181904	SYN Scan (Inbound)

UDP Scan (Outbound)

Severity: Noise
Occurrences: 972,972
Snort source: spp_portscan

Summary: There is an incredible amount of this traffic in relation to the overall scan activity. Out of 1,236,973 scan events 972,972 are outbound udp. I can confirm that 910,000 of these events are either online gaming (Half-Life, Medal of Honor) or peer to peer file sharing (WinMX).

Medal of Honor:
Sep 17 00:01:25 MY.NET.70.207:12203 -> 24.170.50.191:12206 UDP
Sep 17 00:01:25 MY.NET.70.207:12203 -> 24.170.50.191:12204 UDP
Sep 17 00:01:25 MY.NET.70.207:12203 -> 24.170.50.191:12205 UDP
Sep 17 00:01:24 MY.NET.70.207:12300 -> 207.6.166.53:65172 UDP
Sep 17 00:01:27 MY.NET.70.207:12203 -> 24.170.50.191:12205 UDP
Sep 17 00:01:27 MY.NET.70.207:12203 -> 24.170.50.191:12206 UDP
Sep 17 00:01:28 MY.NET.70.207:12203 -> 24.170.50.191:12204 UDP
Sep 17 00:01:26 MY.NET.70.207:12300 -> 128.153.159.55:3014 UDP
Sep 17 00:01:27 MY.NET.70.207:12203 -> 68.154.224.75:4358 UDP
Sep 17 00:01:27 MY.NET.70.207:12300 -> 68.9.112.111:3036 UDP
Sep 17 00:01:29 MY.NET.70.207:12203 -> 24.170.50.191:12205 UDP

Half-Life:
Sep 17 23:03:43 MY.NET.87.44:27021 -> 24.72.23.237:27005 UDP
Sep 17 23:03:43 MY.NET.87.44:27021 -> 24.70.144.192:27005 UDP
Sep 17 23:03:43 MY.NET.87.44:27021 -> 12.243.145.217:43620 UDP
Sep 17 23:03:43 MY.NET.87.44:27021 -> 24.207.238.97:27005 UDP
Sep 17 23:03:43 MY.NET.87.44:27021 -> 68.102.132.218:43621 UDP
Sep 17 23:03:43 MY.NET.87.44:27021 -> 12.111.174.37:25437 UDP
Sep 17 23:03:43 MY.NET.87.44:27021 -> 63.164.242.21:43523 UDP
Sep 17 23:03:43 MY.NET.87.44:27021 -> 68.81.200.26:19594 UDP
Sep 17 23:03:42 MY.NET.87.44:27021 -> 137.112.138.174:2273 UDP
Sep 17 23:03:43 MY.NET.87.44:27021 -> 67.38.16.186:27005 UDP

WinMX:
Sep 18 13:09:44 MY.NET.114.45:6257 -> 66.24.83.223:6257 UDP
Sep 18 13:09:45 MY.NET.114.45:6257 -> 24.159.188.35:6257 UDP
Sep 18 13:09:45 MY.NET.114.45:6257 -> 219.164.135.135:6257 UDP
Sep 18 13:09:46 MY.NET.114.45:6257 -> 65.35.189.172:6257 UDP
Sep 18 13:18:16 MY.NET.114.45:6257 -> 24.61.62.115:6257 UDP
Sep 18 13:18:17 MY.NET.114.45:6257 -> 24.66.41.247:6257 UDP
Sep 18 13:18:16 MY.NET.83.146:6257 -> 24.67.89.180:6257 UDP
Sep 18 13:18:16 MY.NET.83.146:6257 -> 64.253.104.26:6257 UDP
Sep 18 13:18:17 MY.NET.83.146:6257 -> 80.192.53.106:16257 UDP
Sep 18 13:18:17 MY.NET.83.146:6257 -> 61.92.137.106:6257 UDP

I was able to find the 2 top talkers for Half-Life listed on web sites that track Half-Life servers. I used google to find that WinMX is a peer to peer file sharing app that uses udp port 6257 for source and destination.

Recommendations: I recommend that if you are going to allow any outbound udp traffic, as it seems that you are, then you should disable the portscan preprocessor for udp packets originating from MY.NET. You can use the portscan-ignorehosts preprocessor to do this.

UDP Scan (Inbound)

Severity: noise
Occurrences: 63,685
Snort source: spp_portscan

Summary: Most of this traffic is for port 6970. According to Apple this is used for Quicktime streaming video:

"Recommendations for Firewall Administrators
QuickTime follows the conventions of the RTP and RTSP Internet standards to stream media over the web. To enable QuickTime 4 to work properly inside your firewall, please follow the following IETF recommendation:
Open port 554 for RTSP/TCP data.
Open ports 6970 through 6999 (inclusive) for RTP/UDP data."
Sep 17 12:50:32 205.188.228.145:15404 -> 130.85.145.166:6970 UDP
Sep 17 12:50:34 205.188.228.145:9000 -> 130.85.157.101:6970 UDP
Sep 17 12:50:34 205.188.228.145:18848 -> 130.85.184.23:6970 UDP
Sep 17 12:50:34 205.188.228.145:30496 -> 130.85.88.138:6970 UDP
Sep 17 12:50:34 205.188.228.145:10326 -> 130.85.116.68:6970 UDP
Sep 17 12:50:34 205.188.228.145:25372 -> 130.85.83.72:6970 UDP
Sep 17 12:50:34 205.188.228.145:30176 -> 130.85.145.139:6970 UDP
Sep 17 12:50:34 205.188.228.145:22120 -> 130.85.15.222:6970 UDP

Of the 63,685 occurrences 61,632 are for port 6970. I would say that all of this is for streaming video using QuickTime.

Recommendations: I would recommend that you ignore inbound udp 6970. The snort FAQ has information on how to tune your portscan preprocessor.

SYN Scan (Outbound)

Severity: Noise
Occurrences: 17802
Snort source: spp_portscan

Summary: Almost all of this traffic is noise. Of the 17,802 scan events 15,322 are for ports 80, 443, and 6346. I will not go into descriptions of what ports 80 and 443 are used for ☺. Port 6346 is used for Gnutella. These events are users on MY.NET searching for peers to share files with. Here is a sample of the Gnutella traffic:

```
Sep 17 04:59:54 130.85.163.107:2852 -> 142.103.39.141:6346 SYN ******S*
Sep 17 04:59:55 130.85.163.107:2850 -> 213.64.126.218:6346 SYN ******S*
Sep 17 04:59:55 130.85.163.107:2843 -> 199.8.35.162:6346 SYN ******S*
Sep 17 04:59:55 130.85.163.107:2853 -> 217.39.192.30:6346 SYN ******S*
Sep 17 04:59:55 130.85.163.107:2854 -> 68.69.9.114:6346 SYN ******S*
Sep 17 04:59:55 130.85.163.107:2855 -> 217.83.117.128:6346 SYN ******S*
Sep 17 04:59:55 130.85.163.107:2856 -> 68.46.176.143:6346 SYN ******S*
Sep 17 04:59:56 130.85.163.107:2854 -> 68.69.9.114:6346 SYN ******S*
```

Recommendations: I would recommend that ports 80, 443, and 6346 are ignored for outbound syn scanning. The snort FAQ details methods for tuning your portscan preprocessor.

SYN Scan (Inbound)

Severity: Low
Occurrences: 181,904
Snort source: spp_portscan

Summary: Most of these events are real attempts at reconnaissance. The university in that provided the scans to GIAC is registered as owning entire MY.NET.xxx.xxx class B network. When you have such a large and publicly routable address space like this you will see scans such as these every day:

```
Sep 17 02:15:29 213.11.92.241:22 -> 130.85.151.65:22 SYN ******S*
Sep 17 02:15:29 213.11.92.241:22 -> 130.85.151.66:22 SYN ******S*
Sep 17 02:15:29 213.11.92.241:22 -> 130.85.151.70:22 SYN ******S*
Sep 17 02:15:29 213.11.92.241:22 -> 130.85.151.72:22 SYN ******S*
Sep 17 02:15:29 213.11.92.241:22 -> 130.85.151.73:22 SYN ******S*
Sep 17 02:15:29 213.11.92.241:22 -> 130.85.151.61:22 SYN ******S*
Sep 17 02:15:29 213.11.92.241:22 -> 130.85.151.62:22 SYN ******S*
Sep 17 02:15:29 213.11.92.241:22 -> 130.85.151.63:22 SYN ******S*
```

Above we a host in France that is scanning MY.NET for the default SSH port.

```
Sep 17 09:35:02 210.0.212.151:3694 -> 130.85.145.66:21 SYN ******S*
```

```
Sep 17 09:35:02 210.0.212.151:3695 -> 130.85.145.67:21 SYN ******S*
Sep 17 09:35:02 210.0.212.151:3696 -> 130.85.145.68:21 SYN ******S*
Sep 17 09:35:02 210.0.212.151:3697 -> 130.85.145.69:21 SYN ******S*
Sep 17 09:35:02 210.0.212.151:3698 -> 130.85.145.70:21 SYN ******S*
Sep 17 09:35:02 210.0.212.151:3701 -> 130.85.145.73:21 SYN ******S*
Sep 17 09:35:02 210.0.212.151:3702 -> 130.85.145.74:21 SYN ******S*
Sep 17 09:35:02 210.0.212.151:3707 -> 130.85.145.79:21 SYN ******S*
```

Now we see a host in China scanning MY.NET for FTP servers.
I would classify most or all of the inbound TCP SYN scans as being probable attempts to locate vulnerabilities on MY.NET. The question is not whether or not these scans are bad, but what we should do with the scan data once we have it. I would think that though 181,904 is a large number that on some days it could be far higher. Since you know that you will see scans of this nature and cannot stop them, what steps can be taken?

Recommendations: I would recommend that a daily report summarizing the top inbound TCP SYN scan sources be compiled from this data, and using this along with a sample of the scan itself a Security Engineer should determine intent and block the source network. I know that this will create overhead in terms of both man-hours and Self-inflicted network problems, but I do not know of any other way to respond to the large volume of scans that are received on a daily basis from all over the internet. You should not automate this process because you would risk causing a denial of service effect were someone to use randomly spoofed source addresses. I believe that if you do not take steps to make a valid security related use of this scan data then it should be turned off.

Interesting Alerts Related to Trojan/Backdoor Activity

Occurrences	Alert
388	IRC evil - running XDCC
120	Possible trojan server activity
86	Port 55850 udp - Possible myserver activity - ref. 010313-1
63	Port 55850 tcp - Possible myserver activity - ref. 010313-1
56	MYPARTY - Possible My Party infection
47	TFTP - Internal UDP connection to external tftp server

IRC evil - running XDCC

Severity: Medium
Occurrences: 388
Snort signature ID: None

Summary: DCC is a method with IRC to do peer to peer file transfers. XDCC bots are IRC clients that have been configured to connect to IRC channels and display

files that they are allowed to serve. When a request is sent to the XDCC bot the file transfer is initiated directly from the bots system to the requesting IRC clients system. The problem is that while there is some legitimate use of XDCC bots, most of them are loaded onto systems that have been compromised on networks that have high-bandwidth Internet connections. Here is a quote from dslreports.com IRC FAQ:

> Q: What's an XDCC? (#4493)
> A: With IRC in full swing, XDCC bots are common sights in channels these days. An XDCC is a bot that has certain packets uploaded to it. These packets may be anything from the recent game to a good movie. XDCCs are usually r00ted (hacked), and transfer at very high speeds because they are on fast lines.

I can only guess as to what this signature is looking for content wise as snort.org does not list any such signature for XDCC. They do have some excellent signatures related to IRC and even file transfer when someone issues a DCC command, such as Snort signature ID's 1639 and 1640. I would guess that this signature matches on the "XDCC" being in the content of and established connection on port 6667. While this traffic is known to be used for illicit activities it is also used for legitimate ones.

Correlations: I can find no other GCIA practical that mention this exact signature.

Recommendations: I would recommend that well-known IRC ports are blocked both inbound and outbound. It is my opinion that IRC has a level of illicit activity on it that has degraded its potential legitimate use to a point where the risk is to great. It is a hotbed of black-hat control channels and illegal bandwidth stealing file sharing. I say leave it to the hackers and find alternate methods to share files and chat.

Possible Trojan server activity

Severity: High
Occurrences: 120
Snort signature ID: None

Summary: It seems that this signature will alert on a source or destination port of 27374. This is the default port of the popular SubSeven Trojan as well as at least 15 others. Most of the activity that matched this scan are false positives due to the nature of the signature. Below is a sample of an established Kazza peer-to-peer file sharing connection that alerted on this signature.

```
09/19-07:53:46.971662  [**] Possible trojan server activity [**] MY.NET.83.153:1214 ->
166.90.245.97:27374
09/19-07:53:46.971787  [**] Possible trojan server activity [**] MY.NET.83.153:1214 ->
166.90.245.97:27374
09/19-07:53:46.971961  [**] Possible trojan server activity [**] MY.NET.83.153:1214 ->
166.90.245.97:27374
09/19-07:53:53.277966  [**] Possible trojan server activity [**] MY.NET.83.153:1214 ->
```

166.90.245.97:27374
09/19-07:54:00.922433 [**] Possible trojan server activity [**] MY.NET.83.153:1214 -> 166.90.245.97:27374
09/19-07:54:16.211380 [**] Possible trojan server activity [**] MY.NET.83.153:1214 -> 166.90.245.97:27374
09/19-07:54:16.400816 [**] Possible trojan server activity [**] 166.90.245.97:27374 -> MY.NET.83.153:1214

However there is a small amount that indicates a real infection on a MY.NET host:

09/21-14:11:41.197113 [**] Possible trojan server activity [**] 80.200.45.69:1302 -> MY.NET.137.1:27374
09/21-14:11:41.197666 [**] Possible trojan server activity [**] MY.NET.137.1:27374 -> 80.200.45.69:1302
09/21-14:11:41.528392 [**] Possible trojan server activity [**] 80.200.45.69:1744 -> MY.NET.137.2:27374
09/21-14:11:41.710977 [**] Possible trojan server activity [**] 80.200.45.69:1302 -> MY.NET.137.1:27374
09/21-14:11:41.711500 [**] Possible trojan server activity [**] MY.NET.137.1:27374 -> 80.200.45.69:1302
09/21-14:11:41.856400 [**] Possible trojan server activity [**] 80.200.45.69:2192 -> MY.NET.137.3:27374
09/21-14:11:42.207420 [**] Possible trojan server activity [**] 80.200.45.69:1302 -> MY.NET.137.1:27374

inetnum: 80.200.0.0 - 80.200.255.255
netname: BE-SKYNET-ADSL2
descr: ADSL Customers
descr: Skynet Belgium
country: BE
admin-c: JFS1-RIPE
tech-c: PDH16-RIPE
status: ASSIGNED PA
mnt-by: SKYNETBE-MNT
changed: ripe@skynet.be 20011212
source: RIPE

route: 80.200.0.0/15
descr: SKYNETBE-CUSTOMERS
origin: AS5432
notify: noc@skynet.be
mnt-by: SKYNETBE-MNT
changed: noc@skynet.be 20011116
source: RIPE

person: Jean-Francois Stenuit
address: Belgacom Skynet NV/SA
address: Rue Colonel Bourg 124
address: B-1140 Bruxelles
address: Belgium
phone: +32 2 706-1111
fax-no: +32 2 726-9829
e-mail: jfs@skynet.be

nic-hdl: JFS1-RIPE
remarks: ----------
remarks: Network problems to: noc@skynet.be
remarks: Peering requests to: peering@skynet.be
remarks: Abuse notifications to: abuse@skynet.be
remarks: ----------
mnt-by: SKYNETBE-MNT
changed: jfs@skynet.be 19970707
changed: tech@dns.be 19971003
changed: piet@skynet.be 19991210
changed: piet@skynet.be 20000302
source: RIPE

person: Pieterjan d'Hertog
address: Belgacom Skynet sa/nv
address: 2 Rue Carli
address: B-1140 Brussels
address: Belgium
phone: +32 2 706 13 11
fax-no: +32 2 706 13 12
e-mail: piet@skynet.be
nic-hdl: PDH16-RIPE
remarks: ----------
remarks: Network problems to: noc@skynet.be
remarks: Peering requests to: peering@skynet.be
remarks: Abuse notifications to: abuse@skynet.be
remarks: ----------
mnt-by: SKYNETBE-MNT
changed: jfs@skynet.be 19990415
changed: piet@skynet.be 19991210
changed: piet@skynet.be 20000302
changed: piet@skynet.be 20020329
source: RIPE

There is also another alert that indicates that a MY.NET host is being used to control outside trojaned hosts:

09/21-13:39:35.823837 [**] Possible trojan server activity [**] MY.NET.108.26:3372 -> 161.69.201.238:27374
09/21-13:39:35.902407 [**] Possible trojan server activity [**] 161.69.201.238:27374 -> MY.NET.108.26:3372
09/21-13:39:35.902741 [**] Possible trojan server activity [**] MY.NET.108.26:3372 -> 161.69.201.238:27374
09/21-13:39:35.984995 [**] Possible trojan server activity [**] 161.69.201.238:27374 -> MY.NET.108.26:3372
09/21-13:39:35.985083 [**] Possible trojan server activity [**] 161.69.201.238:27374 -> MY.NET.108.26:3372
09/21-13:39:35.985125 [**] Possible trojan server activity [**] MY.NET.108.26:3372 -> 161.69.201.238:27374
09/21-13:39:35.986508 [**] Possible trojan server activity [**] MY.NET.108.26:3372 -> 161.69.201.238:27374

OrgName: Network General Corp.

OrgID: NGC-6

NetRange: 161.69.0.0 - 161.69.255.255
CIDR: 161.69.0.0/16
NetName: NGC
NetHandle: NET-161-69-0-0-1
Parent: NET-161-0-0-0-0
NetType: Direct Assignment
NameServer: DNS1.NAI.COM
NameServer: DNS2.NAI.COM
NameServer: DNS3.NAI.COM
NameServer: DNS4.NAI.COM
Comment:
RegDate: 1992-06-15
Updated: 2002-05-15

TechHandle: NA5-ORG-ARIN
TechName: Network Associates, Inc.
TechPhone: +1-408-988-3832
TechEmail: hostmaster@nai.com

ARIN Whois database, last updated 2002-10-09 19:05
Enter ? for additional hints on searching ARIN's Whois database

Correlations: Todd Beardsley in his GCIA practical of 31 May 2002 also found a subSeven infection on MY.NET. His was more widespread and showed MY.NET systems connecting to other MY.NET systems.

Recommendations: I would recommend that both of the MY.NET systems involved be taken off the network and completely rebuilt. The level of compromise is probably such that you cannot expect to “clean” the system any other way.

Port 55850 udp - Possible myserver activity

Severity: Medium
Occurrences: 86
Snort signature ID: None

Summary: Some of these alerts are false positives. I believe this alert is looking for any packets with a destination port of 55850. There are reply packets to DNS queries which have matched this alert as well as traceroute packets which have as well. However I think that some of these alerts are legitimate and at least one system on MY.NET may be compromised:

09/20-22:50:56.038287 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
61.165.64.14:55850 -> MY.NET.84.249:55330
09/20-22:50:56.075096 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
61.165.64.14:55850 -> MY.NET.84.249:55330
09/20-22:50:56.102728 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
61.165.64.14:55850 -> MY.NET.84.249:55330
09/20-22:50:56.125633 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]

61.165.64.14:55850 -> MY.NET.84.249:55330
09/20-22:50:56.158067 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
61.165.64.14:55850 -> MY.NET.84.249:55330
09/20-22:50:56.374299 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
61.165.64.14:55850 -> MY.NET.84.249:55330
09/20-22:50:56.736300 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
61.165.64.14:55850 -> MY.NET.84.249:55330
09/20-22:50:56.842216 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
61.165.64.14:55850 -> MY.NET.84.249:55330
09/20-22:50:56.844527 [**] Port 55850 udp - Possible myserver activity - ref. 010313-1 [**]
61.165.64.14:55850 -> MY.NET.84.249:55330

inetnum: 61.165.0.0 - 61.165.255.255
netname: CHINANET-SH
descr: CHINANET Shanghai province network
descr: Data Communication Division
descr: China Telecom
country: CN
admin-c: CH93-AP
tech-c: XI5-AP
mnt-by: MAINT-CHINANET
mnt-lower: MAINT-CHINANET-SH
changed: hostmaster@ns.chinanet.cn.net 20001116
status: ALLOCATED PORTABLE
source: APNIC

person: Chinanet Hostmaster
address: No.31 ,jingrong street,beijing
address: 100032
country: CN
phone: +86-10-66027112
fax-no: +86-10-66027334
e-mail: hostmaster@ns.chinanet.cn.net
nic-hdl: CH93-AP
mnt-by: MAINT-CHINANET
changed: hostmaster@ns.chinanet.cn.net 20020814
source: APNIC

person: Wu Xiao Li
address: Room 805,61 North Si Chuan Road,Shanghai,200085,PRC
country: CN
phone: +86-21-63630562
fax-no: +86-21-63630566
e-mail: ip-admin@mail.online.sh.cn
nic-hdl: XI5-AP
mnt-by: MAINT-CHINANET-SH
changed: ip-admin@mail.online.sh.cn 20010510
source: APNIC

There is no traffic outbound from the MY.NET host to indicate that it has received instructions from a myserver DDoS client and is executing them, but I would be concerned just the same.

Correlations: Jeff Zahr in his GCIA practical of 15 November 2001 analyzed these events as being probable false positives caused by the nature of the signature.

Recommendations: I would recommend that close look is had at MY.NET.84.249. While I was able to determine that some of this traffic is false positives, I was unable to easily explain the traffic directed at MY.NET.84.249. While I did not see any related activity from that MY.NET host, this alert still warrants further investigation.

Port 55850 tcp - Possible myserver activity

Severity: Noise
Occurrences: 63
Snort signature ID: None

Summary: All of these alerts are false positives. The traffic is related to Auth, Gnutella, and HTTP. This is another signature written to alert on established packets for a certain port. I have detailed my opinions on this type of signature in my earlier analysis. Here is a sample of the alerts generated by this signature:

09/20-05:51:20.394754 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
MY.NET.6.40:55850 -> 216.177.60.53:113
09/20-05:51:23.755916 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
MY.NET.6.40:55850 -> 216.177.60.53:113
09/20-05:51:30.396476 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
MY.NET.6.40:55850 -> 216.177.60.53:113
09/20-09:23:58.452597 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
MY.NET.110.178:55850 -> 24.31.228.155:6346
09/20-09:24:00.302795 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
MY.NET.110.178:55850 -> 24.31.228.155:6346
09/20-10:49:05.645602 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
128.138.23.21:6346 -> MY.NET.110.178:55850
09/20-10:49:06.594581 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
198.108.95.140:80 -> MY.NET.163.25:55850
09/20-10:49:06.594670 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
MY.NET.163.25:55850 -> 198.108.95.140:80
09/20-19:18:30.082443 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
207.224.14.85:55850 -> MY.NET.29.3:80
09/20-19:18:30.125126 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
207.224.14.85:55850 -> MY.NET.29.3:80
09/20-19:18:30.207375 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
MY.NET.29.3:80 -> 207.224.14.85:55850
09/20-19:18:30.321938 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
207.224.14.85:55850 -> MY.NET.29.3:80
09/20-19:18:30.417822 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
207.224.14.85:55850 -> MY.NET.29.3:80
09/20-19:18:30.419309 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]
207.224.14.85:55850 -> MY.NET.29.3:80
09/21-10:04:05.990019 [**] Port 55850 tcp - Possible myserver activity - ref. 010313-1 [**]

212.71.44.167:55850 -> MY.NET.185.48:6346

Correlations: Tyler Schacht in his GCIA practical of 16 August 2001 analyzes similar packets as being a real compromise. I analyzed the data he used as an example and came to the conclusion that it was a Napster client.

Recommendations: I would recommend that better signatures be deployed related to myserver DDoS activity. Though the amount of false positives in this case was limited, there could have been many more as evidenced by my previous analysis of similar port based signatures.

MYPARTY - Possible My Party infection

Severity: High
Occurrences: 56
Snort signature ID: None

Summary: At first glance this seems to be another false positive, but after further investigation I have found that the all of these alerts were from one MY.NET host destined to 209.151.250.170. This signature probably alerts on any port 80 traffic destined for 209.151.250.170. This signature is very specific and indicates that a MY.NET host has been infected with the BackDoor-FB.svr.gen which is a payload of the W32/Mypaty@MM virus. BackDoor-FB.svr.gen will attempt to connect to http:// 209.151.250.170 and download a command file with instructions for the backdoor that it will set up. Here is a sample of the events:

09/19-10:52:41.612030 [**] MYPARTY - Possible My Party infection [**] MY.NET.153.146:1201 -> 209.151.250.170:80
09/19-10:53:41.622070 [**] MYPARTY - Possible My Party infection [**] MY.NET.153.146:1219 -> 209.151.250.170:80
09/19-10:54:31.610255 [**] MYPARTY - Possible My Party infection [**] MY.NET.153.146:1219 -> 209.151.250.170:80
09/19-10:54:31.610499 [**] MYPARTY - Possible My Party infection [**] MY.NET.153.146:1219 -> 209.151.250.170:80
09/19-10:55:31.615767 [**] MYPARTY - Possible My Party infection [**] MY.NET.153.146:1237 -> 209.151.250.170:80
09/19-10:57:21.620609 [**] MYPARTY - Possible My Party infection [**] MY.NET.153.146:1276 -> 209.151.250.170:80
09/19-10:58:11.616091 [**] MYPARTY - Possible My Party infection [**] MY.NET.153.146:1276 -> 209.151.250.170:80

OrgName: Cyberverse Online
OrgID: CYBO

NetRange: 209.151.224.0 - 209.151.255.255
CIDR: 209.151.224.0/19
NetName: CYBERVERSE
NetHandle: NET-209-151-224-0-1
Parent: NET-209-0-0-0-0

NetType: Direct Allocation
NameServer: NS1.CYBERVERSE.NET
NameServer: NS2.CYBERVERSE.NET
NameServer: NS3.CYBERVERSE.NET
Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate: 1998-03-09
Updated: 1999-09-27

TechHandle: COH3-ORG-ARIN
TechName: Cyberverse Online Hostmaster
TechPhone: +1-310-643-3783
TechEmail: domain@cyberverse.com

ARIN Whois database, last updated 2002-10-09 19:05
Enter ? for additional hints on searching ARIN's Whois database.

Correlations: Michael Wilkinson in his GCIA practical of 9 April 2002 also saw similar traffic. He used Symantec's anti virus research center as his reference and correctly came to the same conclusions.

Recommendations: The MY.NET host that generated these alerts needs to have it's anti virus software updated and its system clock set to current. This virus only attempts connections to http://209.151.250.170 if its system clock is set between the 25th - 29th of January 2002.

TFTP - Internal UDP connection to external tftp server

Severity: High
Occurrences: 47
Snort signature ID: None

Summary: These alerts are all legitimate MY.NET hosts connecting to external TFTP servers. I would be suspect of every one of these connections. I can correlate at least 2 to VERY suspicious activity. The following is one of the suspect alerts, along with a related (I think) packet:

09/20-17:30:11.363104 [**] High port 65535 udp - possible Red Worm - traffic [**]
211.110.11.143:20885 -> MY.NET.152.173:65535
09/20-17:30:43.958156 [**] TFTP - Internal UDP connection to external tftp server [**]
211.110.11.143:69 -> MY.NET.152.173:23212

We see that a host in Korea has sent a udp packet on port 65535 to MY.NET.152.173. Then shortly after we see that MY.NET.152.173 connects to the same host in Korea via TFTP. I think that the MY.NET host has been compromised and the Korean host is controlling it.

Here is another suspect sample:

09/17-11:16:05.817822 [**] TFTP - Internal UDP connection to external tftp server [**] MY.NET.132.23:1044 -> 80.11.207.237:69
09/18-08:18:38.892188 [**] TFTP - Internal UDP connection to external tftp server [**] MY.NET.157.111:1033 -> 80.11.207.237:69
09/18-08:46:45.921753 [**] TFTP - Internal UDP connection to external tftp server [**] MY.NET.157.30:1048 -> 80.11.207.237:69
09/18-10:58:40.869978 [**] TFTP - Internal UDP connection to external tftp server [**] MY.NET.151.190:1126 -> 80.11.207.237:69
09/19-08:52:04.155634 [**] TFTP - Internal UDP connection to external tftp server [**] MY.NET.190.26:1117 -> 80.11.207.237:69

inetnum: 80.11.207.0 - 80.11.207.255
netname: IP2000-ADSL-BAS
descr: BSANN101 Annecy Bloc2
country: FR
admin-c: WITR1-RIPE
tech-c: WITR1-RIPE
status: ASSIGNED PA
remarks: for hacking, spamming or security problems send mail to
remarks: postmaster@wanadoo.fr AND abuse@wanadoo.fr
remarks: for ANY problem send mail to gestionip.ft@francetelecom.com
mnt-by: FT-BRX
changed: gestionip.ft@francetelecom.com 20011011
source: RIPE

route: 80.11.192.0/19
descr: France Telecom
descr: Wanadoo Interactive
remarks: ----------
remarks: For Hacking, Spamming or Security problems
remarks: send mail to abuse@wanadoo.fr ONLY
remarks: ----------
origin: AS3215
mnt-by: RAIN-TRANSPAC
mnt-by: FT-BRX
changed: karim@rain.fr 20020226
source: RIPE

role: Wanadoo Interactive Technical Role
address: WANADOO INTERACTIVE
address: 48 rue Camille Desmoulins
address: 92791 ISSY LES MOULINEAUX CEDEX 9
address: FR
phone: +33 1 58 88 50 00
e-mail: abuse@wanadoo.fr
e-mail: postmaster@wanadoo.fr
admin-c: FTI-RIPE
tech-c: TEFS1-RIPE
nic-hdl: WITR1-RIPE
notify: gestionip.ft@francetelecom.com
mnt-by: FT-BRX
changed: gestionip.ft@francetelecom.com 20010504

changed: gestionip.ft@francetelecom.com 20010912
changed: gestionip.ft@francetelecom.com 20011204
source: RIPE

80.11.207.237 has been scanning MY.NET for various vulnerabilities throughout the 5 day period that my analysis covers, I will be writing more on him later. It looks as though he has found some systems that were vulnerable and compromised them and is now having them transfer files of an unknown nature via TFTP.

Correlations: Michael McDonnell in his GCIA practical of 27 November 2001 states of TFTP "It's an ideal file transfer program to include in a virus or rootkit." He is absolutely correct.

Recommendations: I would recommend that all TFTP traffic is blocked at a firewall or border router. This should include both inbound and outbound traffic. The legitimate use of TFTP has declined to almost nothing in recent years, and should anyone present a valid need then they could be allowed on a case by case basis.

Out of Spec Packets

When I began to work on my part 3 of my GCIA practical I was unable to find 5 days of corresponding oos files for any set of scan and alert files. I wrote to you good folks at GIAC and was informed that I should analyze the oos events from June separately from my scan and alert data. You have since posted current oos data files that correspond with current scan and alert data files. However as I had already begun my paper I did not change files. I will be appending the email correspondence with GIAC at the end of this paper. Thank you for your understanding.

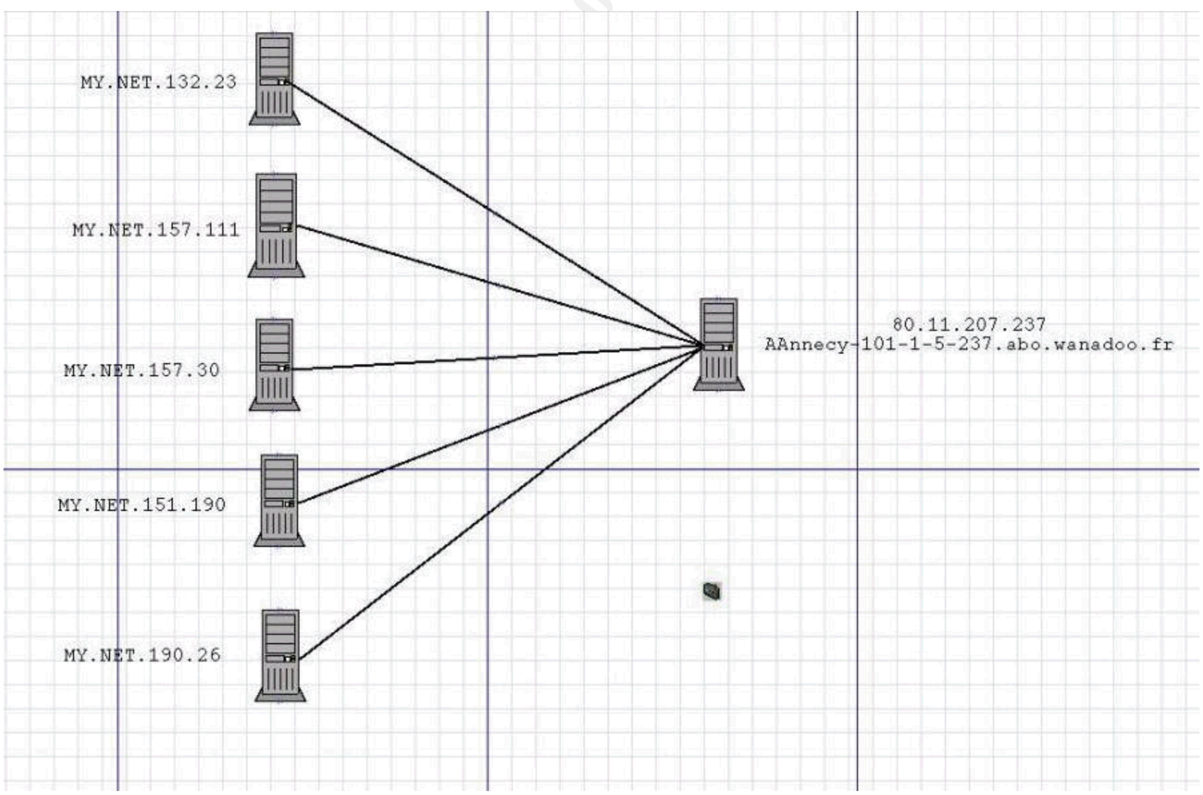
Summary: There were 2998 Out of Spec packets contained in the June oos tarball. 2923 of these are related to the increasing number of devices using the ECN (Explicit Congestion Notification) standard. Basically the standard describes a way to use the 8 and 9 bit in a TCP header for when routers experience congestion. In the past they would just drop the packet and it would eventually get resent, but with ECN routers can send a return packet informing the source that there is congestion. During the TCP three way handshake devices that have implemented the ECN standard will set both the 8 and 9 reserved bits in its SYN, if the destination is also ECN compliant then it will set the 9 bit in its SYN ACK. Toby Miller in his paper describing how ECN will effect intrusion detection outlines the difficulty in distinguishing ECN packets from tools such as Queso which uses the reserved bits for OS fingerprinting. He states that you must analyze more than just the initial SYN packet to determine if this is OS fingerprinting or ECN. Below are some examples of oos packets that look like ECN:

=+
06/18-14:24:45.944943 68.32.126.64:14240 -> MY.NET.6.7:110
TCP TTL:47 TOS:0x0 ID:16001 DF
21S***** Seq: 0x3D06FB4E Ack: 0x0 Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 39765455 0 EOL EOL EOL EOL

=+
06/18-14:25:49.647110 68.32.126.64:14265 -> MY.NET.6.7:110
TCP TTL:47 TOS:0x0 ID:50556 DF
21S***** Seq: 0x40AC1BD5 Ack: 0x0 Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 39771817 0 EOL EOL EOL EOL

=+
06/18-14:26:49.141348 209.116.70.75:56238 -> MY.NET.100.217:25
TCP TTL:51 TOS:0x0 ID:6671 DF
21S***** Seq: 0x51F817A0 Ack: 0x0 Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 757992879 0 EOL EOL EOL EOL

The above look like ECN standard SYN packets described in RFC 2481. I believe that you should eliminate these packets as being Out of Spec. They are within Spec, it is just a relatively new Spec ☺.



Study of 80.11.207.237

The above is a relational graph of the 5 MY.NET hosts who connected via TFTP to 80.11.207.237. That particular destination host has scanned MY.NET for SMB vulnerabilities, and RPC vulnerabilities thousands of times over the 5 day period. It seems as though he was successful in compromising at least these 5 systems. I would block 80.11.207.237 permanently and notify the ISP for that network:

```
inetnum:     80.11.207.0 - 80.11.207.255
netname:      IP2000-ADSL-BAS
descr:       BSANN101 Annecy Bloc2
country:      FR
admin-c:      WITR1-RIPE
tech-c:      WITR1-RIPE
status:      ASSIGNED PA
remarks:      for hacking, spamming or security problems send mail to
remarks:      postmaster@wanadoo.fr AND abuse@wanadoo.fr
remarks:      for ANY problem send mail to gestionip.ft@francetelecom.com
mnt-by:      FT-BRX
changed:      gestionip.ft@francetelecom.com 20011011
source:      RIPE

route:       80.11.192.0/19
descr:       France Telecom
descr:       Wanadoo Interactive
remarks:      ----------------------------------------
remarks:      For Hacking, Spamming or Security problems
remarks:      send mail to    abuse@wanadoo.fr    ONLY
remarks:      ----------------------------------------
origin:      AS3215
mnt-by:       RAIN-TRANSPAC
mnt-by:      FT-BRX
changed:      karim@rain.fr 20020226
source:      RIPE

role:        Wanadoo Interactive Technical Role
address:      WANADOO INTERACTIVE
address:      48 rue Camille Desmoulins
address:      92791 ISSY LES MOULINEAUX CEDEX 9
address:      FR
phone:       +33 1 58 88 50 00
e-mail:      abuse@wanadoo.fr
e-mail:      postmaster@wanadoo.fr
admin-c:      FTI-RIPE
tech-c:      TEFS1-RIPE
nic-hdl:     WITR1-RIPE
notify:      gestionip.ft@francetelecom.com
mnt-by:       FT-BRX
changed:      gestionip.ft@francetelecom.com 20010504
changed:      gestionip.ft@francetelecom.com 20010912
changed:      gestionip.ft@francetelecom.com 20011204
source:      RIPE
```

Top Talkers

Source	Occurrences	Alert

152.163.134.164	41699	SUNRPC highport access!
80.135.86.212	4236	spp http decode: IIS Unicode attack detected
61.155.107.59	3617	Incomplete Packet Fragments Discarded
MY.NET.152.249	3538	spp http decode: IIS Unicode attack detected
212.179.99.36	3511	Watchlist 000220 IL-ISDNNET-990517
MY.NET.153.190	3392	spp http decode: IIS Unicode attack detected
212.179.35.118	2884	Watchlist 000220 IL-ISDNNET-990517
MY.NET.153.170	2299	spp http decode: CGI Null Byte attack detected
MY.NET.85.74	2157	spp http decode: IIS Unicode attack detected
MY.NET.153.174	1983	spp http decode: IIS Unicode attack detected

The above list represents a list of top talkers from the alerts category. As you can see most of the alerts have been earlier identified as being “noise”. This indicates that were you take action to eliminate some of the “noise” on MY.NET you would be able to get a more accurate picture from tables like this one.

Defensive Recommendation/Summary

MY.NET has Snort NIDS infrastructure which gives them an edge when it comes to network security right off the bat. With a little tuning they could have an increased level of visibility without having to compromise security. The biggest problem I am seeing with their overall security posture is that in the constant battle between security and ease of use the latter seems to have one a victory on MY.NET. It is my opinion that they are far to permissive with the services permitted by their Firewall or router access lists. MY.NET has a significant exposure to many common vulnerabilities that are easily blocked by perimeter devices. While I am aware that users are always clamoring for less restrictive access, a balance needs to happen. While I have outline many specific defensive recommendations in my earlier analysis, Other than all of the specific recommendations made inline with my analysis, I think that a good firewall policy is the biggest general defensive recommendation I can make.

Tools used for Analyze this

Microsoft Windows 2000
Microsoft Word XP
Microsoft Word 2000
Mandrake Linux 8.2
Snort 1.8.6
Google (www.google.com)
Geektools (www.geektools.com)
Snort signature/port database (www.snort.org)
SANS Institute (www.sans.org)
Distributed Intrusion Detection System (www.dshield.org)
arachNIDS database (www.whitehats.com/ids/)
Managed SherlockESM (www.lurhq.com/msesm.htm)

I also used the following perl script to parse through the log files. Like most others I combined all 5 days logs into one file each for scans, alerts, and oos and removed the portscan alerts from my alerts data.

```
#!/usr/bin/perl

#09/18-10:03:26.121357  [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.35.118:80 -> MY.NET.153.150:1870
# The above is for reference on how the alerts are formatted

my $count = 0;
while (<>) {
      if (/.*\[\*\*\] (.*) \[\*\*\] (.*) -> (.*)/) {
            $count++;
            my $msg = $1; my $src = $2; my $dst = $3;
            $msg =~ s/[^A-Za-z0-9 -_]//g;
            $msg =~ s/ /_/g;
            open(OUT, ">>msg/$msg");
            print OUT;
            close OUT;
            $src =~ s/:.*//g;
            open(OUT, ">>src/$src");
            print OUT;
            close OUT;
            print "Processed $count lines\n";
      } else {
            print "Skipped $_";
            $skipped .= $_;
      }
}
print "Skipped the following lines:\n$skipped\n";
```

As well as this script I used grep, awk, sort, and uniq extensively. I looked at other GCIA practicals as well as the sites recommended in the student guide, but did not make use of any other scripts or tools. I did not attempt to use SnortSnarf as it seems many people have had problems with it before.

References

"ISS Security Alert" July 12, 1999
http://bvlive01.iss.net/issEn/delivery/xforce/alertdetail.jsp?id=advise31 (October 11, 2002)

BO2K FAQ
http://bo2k.sourceforge.net/docs/faq.html October 10, 2002

Stunnel -- Universal SSL Wrapper
http://www.stunnel.org/ October 11, 2002

Detecting and Containing IRC-Controlled Trojans: When Firewalls, AV, and IDS Are Not Enough *by* Corey Merchant and Joe Stewart, LURHQ Corporation Secure

Operations Center July 10, 2002
http://online.securityfocus.com/infocus/1605 October 11, 2002

CERT® Advisory CA-2002-22 Multiple Vulnerabilities in Microsoft SQL Server July 29, 2002
http://www.cert.org/advisories/CA-2002-22.html October 11, 2002

mIRC FAQ
http://www.mirc.com/faq.html October 11, 2002

.mixter security
http://mixter.warrior2k.com/ October 11, 2002

Entercept Security technologies
http://www.entercept.com/ October 11, 2002

Microsoft IIS 4.0 / 5.0 vulnerable to directory traversal via extended unicode in url (MS00-078) 11/20/2000
http://www.kb.cert.org/vuls/id/111677 October 11, 2002

WEB-IIS cmd.exe access Snort signature
http://www.snort.org/snort-db/sid.html?sid=1002 October 11, 2002

WEB-IIS CodeRed v2 root.exe access Snort signature
http://www.snort.org/snort-db/sid.html?sid=1256 October 11, 2002

WEB-IIS Unicode2.pl script (File permission canonicalization) Snort signature
http://www.snort.org/snort-db/sid.html?sid=989 October 11, 2002

Beardsley, Tod Intrusion Detection and Analysis: Theory, Techniques, and Tools *May 8, 2002*
http://www.giac.org/practical/Tod_Beardsley_GCIA.doc October 11, 2002

arachNIDS
http://www.whitehats.com/ids/ October 11, 2002

Timm, Kevin GCIA Version 3.1 31 May, 2002
http://www.giac.org/practical/Kevin_Timm_GCIA.doc October 11, 2002

Snort Users mailing list archive Stewart, Joe Nov 20 2000
http://archives.neohapsis.com/archives/snort/2000-11/0244.html October 11, 2002

Snort FAQ
http://www.snort.org/docs/faq.html October 11, 2002

Bugtraq post of NT frag attack code

http://www.insecure.org/sploits/NT.no_first_fragment.IP_frag.attack.htm October 11, 2002

Peck, Edward SANS GIAC GCIA Practical Version 3.0 25 November 2001
http://www.giac.org/practical/Edward_Peck_GCIA.doc October 11, 2002

Microsoft Windows Server Message Block (SMB) fails to properly handle SMB_COM_TRANSACTION packets requesting NetServerEnum2 transaction 08/22/2002
http://www.kb.cert.org/vuls/id/250635 October 11, 2002

Microsoft Windows Server Message Block (SMB) fails to properly handle SMB_COM_TRANSACTION packets requesting NetServerEnum3 transaction 08/22/2002
http://www.kb.cert.org/vuls/id/311619 October 11, 2002

Microsoft Windows Server Message Block (SMB) fails to properly handle SMB_COM_TRANSACTION packets requesting NetShareEnum transaction 08/22/2002
http://www.kb.cert.org/vuls/id/342243 October 11, 2002

CERT® Incident Note IN-2002-04 May 22, 2002
http://www.cert.org/incident_notes/IN-2002-04.html October 11, 2002

Firewalls and Quicktime
http://www.apple.com/quicktime/resources/qt4/us/proxy/ October 11, 2002

Instructions on Cleaning IRC bot & backdoor: XDCC
http://security.duke.edu/cleaning/xdcc.html October 11, 2002

dslreports.com IRC FAQ
http://www.dslreports.com/faq/4493 October 11, 2002

CHAT IRC DCC chat request Snort signature
http://www.snort.org/snort-db/sid.html?sid=1640 October 11, 2002

CHAT IRC DCC file transfer request Snort signature
http://www.snort.org/snort-db/sid.html?sid=1639 October 11, 2002

Ports used by trojans (2001-06-30) Joakim von Braun
http://www.simovits.com/nyheter9902.html October 11, 2002

Zahr, Jeff Certification (GCIA) Version 3.0 November 15, 2001
http://www.giac.org/practical/Jeff_Zahr_GCIA.doc October 11, 2002

Schact, Tyler GCIA Practical Assignment August 16, 2001
http://www.giac.org/practical/Tyler_Schacht_GCIA.doc October 11, 2002

BackDoor-FB.svr.gen McAfee AVERT 01/28/2002
http://vil.nai.com/vil/content/v_99333.htm October 11, 2002

Intrusion Detection in Depth: GCIA Practical Assignment Version 3.0
November 27, 2001
http://www.giac.org/practical/Michael_McDonnell_GCIA.doc October 11, 2002

K. Ramakrishnan and S. Floyd A Proposal to add Explicit Congestion Notification (ECN) to IP January 1999
http://www.ietf.org/rfc/rfc2481.txt?number=2481 October 11, 2002

Miller, Toby ECN and it's impact on Intrusion Detection
http://www.sans.org/y2k/ecn.htm October 11, 2002

Wilkinson, Michael GCIA Practical for SANS Darling Harbour
http://www.giac.org/practical/michael_wilkinson_gcia.doc October 11, 2002

Mogul, Jeffrey Broadcasting Internet Datagrams October 1984
http://www.faqs.org/rfcs/rfc919.html