Global Information Assurance Certification Paper

Lone Star SANS 2001 GCIA Practical Version 2.8

Donald Pitts

May 29, 2001

Assignment 1 - Network Detects

Detect #1

```
20:16:51.130820 MY.NET.192.251.3934 > MY.NET.192.180.22:  [udp sum ok] udp 2 (ttl 127, id 747, len 30)
0x0000   4500 001e 02eb 0000 7f11 aef0 .... c0fb        E.............."..
0x0010   .... c0b4 0f5e 0016 000a 1821 4e51 2020        ."...^.....!NQ..
0x0020   2020 2020 2020 2020 2020 2020 2020 723f        ..............r?
0x0030   9984                                           ..
20:17:18.137748 MY.NET.192.251.3946 > MY.NET.192.180.5632:  [udp sum ok] udp 2 (ttl 127, id 16366, len 30)
0x0000   4500 001e 3fee 0000 7f11 71ed .... c0fb        E...?.....q.."..
0x0010   .... c0b4 0f6a 1600 000a 022b 4e51 2020        ."...j.....+NQ..
0x0020   2020 2020 2020 2020 2020 2020 2020 11c6        ................
0x0030   d1af                                           ..
20:17:18.142026 MY.NET.192.251.3946 > MY.NET.192.180.22:  [udp sum ok] udp 2 (ttl 127, id 16622, len 30)
0x0000   4500 001e 40ee 0000 7f11 70ed .... c0fb        E...@.....p.."..
0x0010   .... c0b4 0f6a 0016 000a 1815 4e51 2020        ."...j......NQ..
0x0020   2020 2020 2020 2020 2020 2020 2020 f923        ...............#
0x0030   71ae                                           q.
```

1. Source of Trace:

A home DSL network with a single dynamically assigned IP address to a firewall router protecting a Windows PC. Shadow setup on a Solaris 7 x86 box with the interface down (no IP address) collected the traffic connected to a hub outside the firewall router. The only traffic arriving at this sensor are the packets addressed to this one IP address.

2. Detect was generated by:

Examining by hand unexpected traffic during an otherwise quiet Internet connection. The detect is presented in tcpdump verbose format with the packet body shown in both hex and ASCII.

3. Probability the source address was spoofed:

Relatively low. It could be spoofed, but this appears to be a query to see whether some services are available which would require a response for it to be useful. The TTL is 127 which means it very likely was 128 and this is only one hop away. That corresponds fairly well with the address being in the same class C network. If the packet was crafted, then the TTL could have been set to make it look this way on purpose.

4. Description of the attack:

This is a probe against UDP ports 22 and 5632 from an address within the same class C network as the target with "NQ" as the application data.

This probe does not relate to a CVE directly, but should the probe be successful there are three CVEs that could be applicable (http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=pcanywhere):

Name	Description
CVE-2000-0273	PCAnywhere allows remote attackers to cause a denial of service by terminating the connection before PCAnywhere provides a login prompt.
CAN-2000-0300	The default encryption method of PcAnywhere 9.x uses weak encryption, which allows remote attackers to sniff and decrypt PcAnywhere or NT domain accounts.
CAN-2000-0324	pcAnywhere 8.x and 9.x allows remote attackers to cause a denial of service via a TCP SYN scan, e.g. by nmap.

5. Attack mechanism:

This probe is querying for an agent of the Symantec PCAnywhere product to respond by delivering an application payload of two bytes containing "NQ" to two of the well known ports for this product. PCAnywhere is a program which runs to provide remote access to a PC. The capability could be thought of as comparable to remotely obtaining a CDE desktop for a UNIX machine. A window is provided on the remote machine containing the display of the other PC. Applications can be run and files can be transferred between the local and remote PCs. Password protection can be configured.

PCAnywhere versions 2.0, 7.0, 7.50, 7.51, and CE use ports TCP 65301 (for data) and UDP 22 (for status) by default. With versions 7.52 and newer the default ports TCP 5631 (for data) and UDP 5632 (for status) is used by default now (http://service1.symantec.com/SUPPORT/pca.nsf/docid/1998122810210812). Versions 8.x and 9.0 both automatically connect with the older ports as well for backward compatiblity. With versions 9.2 and more recent, Symantec will no longer support the old port numbers.

Symantec is requesting all customers alter the configuration of their older copies to use the newer ports as well to make it comply with the port assignemts of the Internet Assigned Numbers Authority (IANA) (http://service1.symantec.com/SUPPORT/pca.nsf/docid/2000071816525412).

An option is provided within PCAnywhere that searches the local Class C network for other PCAnywhere agents rather than configuring the specific IP addresses which may be legitimately reachable. A list of the PCAnywhere capable machines located is then shown from which a particular machine can be picked.

A starting TTL of 128 is usually used by Windows-based machines according to http://project.honeynet.org/papers/finger/traces.txt and that would be necessary for this to be a legitimately running PC Anywhere as opposed to using some hacker tool on a non-Windows box. Of course, they could be running a hacker tool on a Windows box and then this wouldn't distinguish between a hapless user and a hacker.

At first this appeared this might be some sort of SSH probe because the nmap-services file with nmap indicates SSH on both TCP and UDP. Investigation showed SSH to only be a TCP only based service which lead to further searching to discover the PCAnywhere connection.

6. Correlations:

No specific complaints or similar traffic was discovered on the Internet regarding this specific source IP address. This would continue to correspond with the "attacker" being a normal user. There were relevant discussions of PC Anywhere probes in general, however:

Post by Joel Colvin - http://www.netsys.com/firewalls/firewalls-2000-03/msg00257.html
Post by Lenny Zeltser - http://www.sans.org/y2k/123199-1220.htm
Robert Graham - "Firewall Forensics FAQ" - http://www.robertgraham.com/pubs/firewall-seen.html

My first indication this was almost certainly PCAnywhere traffic was when I found "How to Handle and Identify
Network Probes" by Ron Gula at http://packetstorm.securify.com/docs/infosec/probes.txt which contains a few sentences discussing the connection between UDP packets addressed to port 22 with "NQ" ASCII data in the payload with PCAnywhere.

A good page describing the workings of this traffic is at:
http://advice.networkice.com/Advice/Intrusions/2001507/default.htm

The backdoor-lib signature file that comes with Snort has an alert to flag the probe of UDP port 22 with the "NQ" application payload:

```
alert udp any any -> $HOME_NET 5632 (msg:"PCAnywhere"; content:"ST";)
alert udp any any -> $HOME_NET 22 (msg:"PCAnywhere"; content:"ST";)
alert udp any any -> $HOME_NET 22 (msg:"PCAnywhere"; content:"NQ";)
```

The misc-lib signature file from Snort also has some PC Anywhere related alert messages:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 5631 (msg:"MISC-PCAnywhere Attempted Administrator
Login";flags:PA;content:"ADMINISTRATOR";)
alert udp $EXTERNAL_NET any -> $HOME_NET 5632 (msg:"IDS239 - MISC-PCAnywhere Startup";content:"ST"; depth: 2;)
alert tcp $HOME_NET 5632 -> $EXTERNAL_NET any (msg:"IDS240 - MISC-PCAnywhere Failed Login";flags:PA;content:"Invalid login"; depth: 16;)
```

One odd observation to mention is that the Snort rules do not cover the case seen by this traffic where the "NQ" application content is sent to UDP port 5632. Either the traffic seen was not legitimate, or the Snort rules do not cover all of the cases.

Some interesting discussions relating to efforts to get Symantec to change the class C probing is available at http://www.august.net/pcanywhere.html.

7. Evidence of active targeting:

It is not likely this is targeted. This network is not running PCAnywhere and does not have the probed ports open. If this is legitimate, then most likely all of MY.NET.192 and if not, then likely more even more systems got probed.

8. Severity:

Severity = (Criticality [1] + Lethality [5]) - (System [3] + Network [4] Countermeasures) = -1

Criticality - 1 - This is a personal PC that I would not like to lose, but there is nothing overly important that couldn't be replaced.
Lethality - 5 - If PC Anywhere was running and vulnerable, then this would be high.
Countermeasures
System - 3 - This service is not running on the system and it is reasonably recent in its patches, but it is Windows.
Network - 4 - The firewall router is blocking externally initiated traffic with NAT.

9. Defensive recommendation:

Avoid running PCAnywhere and similar products if at all possible. If not, consider blocking PCAnywhere with a dedicated or personal firewall and limiting its usage to internal of your network. Perhaps this traffic could be restricted to a VPN if it must transit the Internet. Regardless, the network query capability should be disabled to avoid announcing oneself as a potential target as well as a common courtesy for those on your immediate network.

10. Multiple choice test question:

Over what size network will Symantec PCAnywhere automatically send probes to all hosts if the user does not configure it to contact particular hosts directly?
a) class A
b) class B
c) class C
d) class D

Answer: c

Detect #2

April 29, 2001:

```
11:31:52.666895 133.91.80.42.1540 > MY.NET.192.180.53:  [udp sum ok] 45694+ [b2&3=0x180] TXT CHAOS)? version.bind. [|domain] (ttl 43, id
42025, len 58)
11:31:52.708426 MY.NET.192.180.1024 > MYDNS.NET.2.1.53:  [udp sum ok] 21+ [b2&3=0x180] TXT CHAOS)? version.bind. [|domain] (DF) (ttl 255,
id 18026, len 58)
11:31:52.735117 MYDNS.NET.2.1.53 > MY.NET.192.180.1024:  [udp sum ok] 21* q: TXT CHAOS)? version.bind. 1/0/0 VERSION.BIND. CHAOS) TXT
8.2.3-REL (64) (DF) (ttl 250, id 9993, len 92)
11:31:52.737329 MY.NET.192.180.53 > 133.91.80.42.1540:  [udp sum ok] 45694* q: TXT CHAOS)? version.bind. 1/0/0 VERSION.BIND. CHAOS) TXT
8.2.3-REL (64) (DF) (ttl 255, id 18027, len 92)
11:31:52.948845 133.91.80.42 > MY.NET.192.180: icmp: 133.91.80.42 udp port 1540 un
reachable for MY.NET.192.180.53 > 133.91.80.42.1540:  [udp sum ok] 45694* q: TXT CHAOS)? version.bind. 1/0/0 VERSION.BIND. CHAOS) TXT
8.2.3-REL (64) (DF) (ttl 234, id 18027, len 92) [tos 0xc0]  (ttl 43, id 42550, len 120)
```

April 29, 2001:

```
11:53:53.545123 133.91.80.42.1960 > MY.NET.192.180.53:  [udp sum ok] 46798+ [b2&3=0x180] TXT CHAOS)?
version.bind. M-^G^B^J (30) (ttl 43, id 20709, len 58)
11:53:53.582860 MY.NET.192.180.1024 > MYDNS.NET.2.1.53:  [udp sum ok] 22+ [b2&3=0x180] TXT CHAOS)?
version.bind. M-I=M-  (30) (DF) (ttl 255, id 18073, len 58)
11:53:53.608959 MYDNS.NET.2.1.53 > MY.NET.192.180.1024:  [udp sum ok] 22* q: TXT CHAOS)? version.bind. 1/0/0
VERSION.BIND. CHAOS) TXT 8.2.3-REL (64) (DF) (ttl 250, id 20213, len 92)
11:53:53.611124 MY.NET.192.180.53 > 133.91.80.42.1960:  [udp sum ok] 46798* q: TXT CHAOS)? version.bind. 1/0/0
VERSION.BIND. CHAOS) TXT 8.2.3-REL (64) (DF) (ttl 255, id 18074, len 92)
11:53:53.828393 133.91.80.42 > MY.NET.192.180: icmp: 133.91.80.42 udp port 1960 unreachable for
MY.NET.192.180.53 > 133.91.80.42.1960:  [udp sum ok] 46798* q: TXT C
HAOS)? version.bind. 1/0/0 VERSION.BIND. CHAOS) TXT 8.2.3-REL (64) (DF) (ttl 234, id 18074, len 92) [tos 0xc0]
(ttl 43, id 21179, len 120)
```

April 30, 2001:

```
02:34:01.082449 133.34.34.76.1960 > MY.NET.192.180.53:  [udp sum ok] 40522+ [b2&3=0x180] TXT CHAOS)?
version.bind. <ELT 35>[|domain] (ttl 46, id 50609, len 58)
02:34:01.084876 MY.NET.192.180.1024 > MYDNS.NET.2.1.53:  [udp sum ok] 60+ [b2&3=0x180] TXT CHAOS)?
version.bind. <ELT 43>[|domain] (DF) (ttl 255, id 20175, len 58)
02:34:01.110845 MYDNS.NET.2.1.53 > MY.NET.192.180.1024:  [udp sum ok] 60* q: TXT CHAOS)? version.bind. 1/0/0
VERSION.BIND. CHAOS) TXT 8.2.3-REL (64) (DF) (ttl 250, id 6117, len 92)
02:34:01.113047 MY.NET.192.180.53 > 133.34.34.76.1960:  [udp sum ok] 40522* q: TXT CHAOS)? version.bind. 1/0/0
VERSION.BIND. CHAOS) TXT 8.2.3-REL (64) (DF) (ttl 255, id 20176, len 92)
02:34:01.342644 133.34.34.76 > MY.NET.192.180: icmp: 133.34.34.76 udp port 1960 unreachable for
MY.NET.192.180.53 > 133.34.34.76.1960:  [udp sum ok] 40522* q: TXT CHAOS)? version.bind. 1/0/0 VERSION.BIND.
CHAOS) TXT 8.2.3-REL (64) (DF) (ttl 239, id 20176, len 92) [tos 0xc0]  (ttl 46, id 50629, len 120)
```

1. Source of Trace:

A home DSL network with a single dynamically assigned IP address to a firewall router protecting a Windows PC. Shadow setup on a Solaris 7 x86 box with the interface down (no IP address) collected the traffic connected to a hub outside the firewall router.

experiment1.phys.chuo-u.ac.jp (133.91.80.42), Netname: JAPAN-INET, Netblock: 133.0.0.0 - 133.255.255.255, Japan Network Information Center (NETBLK-JAPANB-INET)

random.osu.seg.ynu.ac.jp (133.34.34.76), Netname: JAPAN-INET, Netblock: 133.0.0.0 - 133.255.255.255, Japan Network Information Center (NETBLK-JAPANB-INET)

2. Detect was generated by:

Examining by hand unexpected traffic during an otherwise quiet Internet connection. The detect is presented in tcpdump verbose format. The number 45694 following "[udp sum ok]" corresponds to a DNS ID number which should match between the query and response. The plus sign appended to the end of this number indicates that the Recursion Desired (RD) flag was set. The "[b2&3=0x180]" string is shown because there is something odd within the 16 bit DNS Flags field corresponding to bytes 2 and 3 of the DNS message format.The next string of "TXT" is the query type and "CHAOS" is the query class.

3. Probability the source address was spoofed:

It seems very possible based upon the ICMP port unreachable messages being returned by the machines that supposedly originally sent the queries. There is some possibility that these messages are being sent as part of a larger Denial of Service (DoS) attack against the DNS server, however it seems likely we would see more than just a few of these due to the hassle with finding enough susceptible DNS clients to otherwise jam up the DNS server. It is also possible that the originator is located close to the target or the spoofed source address and it sniffing for the responses.

4. Description of the attack:

A UDP packet was sent to port 53 and contained a recursive version.bind query to the chaos query class of a text strings query type. The packet has the Recursion Available (RA) field is set as well. A home firewall router is at this address and it appears that it forwarded the request on to the DNS server. There are a large numbe of CVEs for BIND, but nothing seems obviously of direct applicability.

5. Attack mechanism:

The "b2&3" field corresponds to bytes 2 and 3 of the DNS header and contain a value of 0x180 which is worth investigating further. The first byte is set to "1" which means this is a recursive query. The "80" is not normal or valid. The upper bit of this byte is set, but should always be unset. So this byte should have been "00" and then the "b2&3" field would not have been printed. The first message is reported as being truncated "[|domain]". I believe because of the fact that there should be an answer since the RA flag is set. There are 30 bytes for the DNS query and 8 bytes in the UDP header which totals to 38 matching the length in the UDP header, so it does not appear to be truncated. It appears that the RA flag is being set as an effort to trick answering processes to fulfill the recursive requests or to circumvent filtering/logging rules.

The query is in fact being relayed on to the actual DNS server, a response is received with the version of BIND that is being run, and a similar response is forwarded back

to the original requestor. Suprisingly an ICMP port unreachable message is returned which indicates that no service is listening on the ephemeral source port indicated in the original request, but that the machine itself does exist and is running. Since the response was not obviously processed, it is not apparent whether the intent was to actually retrieve the version of BIND running (`8.2.3-REL`, perform a DoS on my ISP's DNS server, or some other undiscovered motive.

6. Correlations:

http://www.incidents.org/cid/search_result.php?source=133.91.80.42&

Date	Source	Source Port	Target Port	Protocol	Flags
2001-04-29	133.91.80.42	0	53	0	
2001-04-29	133.91.80.42	0	53	6	
2001-04-29	133.91.80.42	0	53	17	

http://www.incidents.org/cid/search_result.php?source=133.34.34.76&

Date	Source	Source Port	Target Port	Protocol	Flags
2001-04-29	133.34.34.76	0	53	0	
2001-04-30	133.34.34.76	0	53	6	
2001-04-30	133.34.34.76	0	53	17	

Similar traffic is shown a post by Andrew: http://www.isc.org/ml-archives/bind-users/1999/11/msg01150.html

16:02:37.481744 146.83.22.169.1051 > 128.61.my.host.53: 3587+ [b2&3=0x180] TXT CHAOS)? version.bind. (30)
16:02:37.481744 128.61.my.host.53 > 146.83.22.169.1051: 3587* 1/0/0 CHAOS) TXT 8.1.2 (60)

7. Evidence of active targeting:

Based upon the correlations on incidents.org, this does not appear to be targeted toward this particular host. It appears that many hosts are likely all being scanned from this particular attacker, or this is part of some bigger DoS attempt.

8. Severity:

Severity = (Criticality [1] + Lethality [3]) - (System [3] + Network [1] Countermeasures) = 0

Criticality - 1 - This is a personal PC that I would not like to lose, but there is nothing overly important that couldn't be replaced.
Lethality - 3 - This appears to be reconnaissance to build a map of who is running which versions of BIND. If my provider's DNS server is compromised, at the least my service may be unavailable or the DNS responses I receive may contain evil misinformation.
Countermeasures
System - 3 - This system is only a DNS client and does not appear to be involved. It is reasonably recent in its patches, but it is Windows.
Network - 1 - The firewall router is facilitating the queries and actively providing answers!

9. Defensive recommendation:

Block incoming DNS UDP queries if possible since this network is not running DNS locally. BIND is not being run in this case, but you should upgrade to the latest version if it was (http://www.cert.org/advisories/CA-2001-02.html). Any BIND server should be configured to severely restrict, if not outright prohibit, the release of version information.

10. Multiple choice test question:

What does "b2&3" from "[b2&3=0x180]" on a DNS line within tcpdump refer to?
a) DNS Flags field
b) Bytes 2 and 3 of the IP header
c) Bytes 2 and 3 of the UDP header
d) Best two out of three ain't bad

Answer: a

Detect #3

23 May 01 10:25:52 tcp 202.152.1.27.61281 o> 202.37.88.81.20002 s
23 May 01 10:25:52 tcp 202.152.1.27.61281 o> 202.37.88.84.20002 s
23 May 01 10:25:52 tcp 202.152.1.27.61281 o> 202.37.88.86.20002 s
23 May 01 10:25:52 tcp 202.152.1.27.61281 o> 202.37.88.88.20002 s
23 May 01 10:25:52 tcp 202.152.1.27.61281 o> 202.37.88.90.20002 s
23 May 01 10:25:52 tcp 202.152.1.27.61281 o> 202.37.88.92.20002 s
23 May 01 10:25:52 tcp 202.152.1.27.61281 o> 202.37.88.94.20002 s
23 May 01 10:25:52 tcp 202.152.1.27.61281 o> 202.37.88.96.20002 s
23 May 01 10:25:52 tcp 202.152.1.27.61281 o> 202.37.88.98.20002 s
23 May 01 10:25:52 tcp 202.152.1.27.61281 o> 202.37.88.101.20002 s
23 May 01 10:25:52 tcp 202.152.1.27.61281 o> 202.37.88.103.20002 s
23 May 01 10:25:52 tcp 202.152.1.27.61281 o> 202.37.88.105.20002 s

1. Source of Trace:

The trace from an Indonesian address shown above was posted on incidents.org on May 23, 2001.
http://www.incidents.org/archives/intrusions/msg00402.html

```
% Rights restricted by copyright. See http://www.apnic.net/db/dbcopyright.html
inetnum:    202.152.1.24 - 202.152.1.31
netname:    IDOLA-DED-UMY
descr:      Dedicated Lan UMY
country:    ID
admin-c:    CT1-ID
tech-c:     YA1-AP
mnt-by:     MAINT-LINTASARTA
changed:    boby@idola.net.id 981004
source:     APNIC
```

2. Detect was generated by:

Unknown. Source is not stated and format of log is not familiar. While the specific format is not known, its contents seem fairly obvious. Each line corresponds to a packet and shows date, time, protocol (tcp), source IP address (202.152.1.27), a period, source port number (61281), an arrow indicating the source is on the left and the destination is on the right, destination IP address (202.37.88.81), a period, destination port number (20002), and TCP flags (s = SYN).

3. Probability the source address was spoofed:

Low. These messages most likely intend to find a process listening on the port probed and hearing back any answer is critical. There are not any likely or easy ways to facilitate this short of using the proper source IP address.

4. Description of the attack:

SYN packets are being sent very quickly to a single destination port (20002) on many machines within the class C network 202.37.88.0. The addresses are being probed in ascending order skipping one or two IP addresses each time.

Common Vulnerabilities and Exposures (CVE):

Name	Description
CAN-1999-0660	A hacker utility or Trojan Horse is installed on a system, e.g. NetBus, Back Orifice, Rootkit, etc.

5. Attack mechanism:

The attacker is sending a SYN packet to the targets and hoping to get a responding SYN ACK indicating that the port is available and listening. A trojan named AcidkoR can only listen on TCP port 20002 and is the likely target of these probes. This trojan runs on Windows 95/98/ME machines. This trojan is a Remote Administration Tool (RAT) which would permit someone to control the PC remotely. The trojan uses telnet on the client side to control it. More information is available at http://www.simovits.com/trojans/tr_data/y33.html and http://www.dark-e.com/archive/trojans/acidkor/index.shtml.

6. Correlations:

This source address has been noted as probing the TCP RPC service on May 18, 2001 with 23 packets against 12 target addresses.
http://www1.dshield.org/ipinfo.php?ip=202.152.1.27

7. Evidence of active targeting:

There is not much evidence to go on. We don't know about the network being protected. It appears this is a brute force scan to find an existing trojan. It would seem likely that they scanned other networks as well.

8. Severity:

Severity = (Criticality [1] + Lethality [5]) - (System [4] + Network [3] Countermeasures) = -1

Criticality - 1 - There could be Window-based machines within this network, but they wouldn't likely be running critical services.
Lethality - 5 - The attacker would likely be able to control any box that responded to this probe.
Countermeasures
System - 4 - We have no information about the systems being protected, but we will assume that they are patched well.
Network - 3 - We have litle information about the network and are not even sure if the packets in question made it to their destination.

9. Defensive recommendation:

Ensure that network devices are in place to block such packets and, more preferably, only permit the necessary services. Running virus protection and a personal firewall on Windows 95/98/ME machines.

10. Multiple choice test question:

Which type of machine would be necessary for an attacker to utilize to access a machine running the AcidkoR trojan?
a) Linux
b) Windows 95/98/ME
c) None - cannot be accessed over the network

d) Just about any network capable platform

Answer: d

Detect #4

```
[**] SCAN-SYN FIN [**]
04/28-22:51:01.425831 192.43.163.81:21 -> MY.NET.192.180:21
TCP TTL:21 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x5A86BCC0  Ack: 0x4E40819D  Win: 0x404  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SCAN-SYN FIN [**]
04/28-23:03:55.809807 192.43.163.81:53 -> MY.NET.192.180:53
TCP TTL:21 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x4B430AAF  Ack: 0x29B6AD0E  Win: 0x404  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

1. Source of Trace:

A home DSL network with a single dynamically assigned IP address to a firewall router protecting a Windows PC. Shadow setup on a Solaris 7 x86 box with the interface down (no IP address) collected the traffic connected to a hub outside the firewall router.

> TELEVERKET Prh (NETBLK-TVTLAN-NETS)
> S-12386 Farsta
> SE
>
> Netname: TVTLAN-NETS
> Netblock: 192.43.162.0 - 192.43.171.255
> Maintainer: TVER
>
> Coordinator:
>   Hellman, Anders (AH96-ARIN) anders.hellman@DATA.TELIA.SE
>   46 8 707 1805

2. Detect was generated by:

Snort.was run reading tcpdump binary files already collected. The first two entries are from the alert logs for this particular source address. The corresponding Snort rule from the scan-lib file looks for any TCP messages from an external address to our network with SYN and FIN flags set which is not a normal or valid packet. The first line identifies the corresponding rule generating the alert log entry. The next line contains the date, time, source IP address (192.43.163.81), source port number (21), destination IP address (MY.NET.192.180), and destination port number (21). The next line indicates the protocol (TCP), Time-To-Live (21), Type Of Service (0x0), IP Fragment ID (39426), IP Header Length (20), and IP Datagram Length (40). The last line for an entry has the TCP flags (SYN and FIN = "******SF"), TCP Sequence Number (0x5A86BCC0), TCP Acknowledgement Number (0x4E40819D), TCP Window Size (0x404), and TCP Header Length (20).

3. Probability the source address was spoofed:

This appears to be a probe which would expect a response, so it seems unlikely that this is spoofed. The SYN and FIN flags would not be combined in normal traffic ever, so this is very clear evidence of crafted packets. If the prober has a sniffer near the source or destination addresses, then they could potentially monitor for a response with the source address spoofed, but this is not very likely.

4. Description of the attack:

TCP packets with SYN and FIN flags set and the source and destination ports the same arrived at both the ftp and DNS ports.
The next thing of interest that comes to light is that the ID on both packets is the same (39426)! This is very unexpected as this should be different between each packet since this is used to identify common IP fragments belonging to the same packet. The window size of 0x404 is 1028 in decimal.

5. Attack mechanism:

The tool used is likely SynScan by psychoid according to research. All of the characteristics correspond. Comments in the source code says it originated somewhat from sscan. If it receives a SYN ACK back, then it sends a RST to close the connection and then sends a normal SYN to begin a normal connection. Guy Bruneau discusses SynScan and indicates that the TTL starts at 42 (http://www.sans.org/y2k/112700-1400.htm and http://archives.neohapsis.com/archives/snort/2000-11/0335.html). The tool is supposed to be available at
http://www.psychoid.lam3rz.de/synscan.html, but it appears with a cursory examination of the source code that the current 1.6 version no longer has many of the characteristics indicated here.

Early speculation as to the origins of this tool can be found at: http://www.shmoo.com/mail/ids/nov00/msg00045.shtml and http://superbofh.org/idlescan.

6. Correlations:

http://www.incidents.org/cid/search_result.php?source=192.43.163.81&

Date	Source	Source Port	Target Port	Protocol	Flags
2001-04-28	192.43.163.81	0	53	0	
2001-04-29	192.43.163.81	0	53	6	
2001-04-29	192.43.163.81	0	21	6	

2001-04-30	192.43.163.81	0	21	6	
2001-05-02	192.43.163.81	21	21	6	S

Traffic just like this can be found at: http://certworks.net/ids/data/snfout.snort_portscan.log/210/174/164/dest210.174.164.132-2401.html

```
[**] IDS198 - SCAN-SYN FIN [**]
 02/01-02:09:29.522635 212.104.15.198:21-> 210.174.164.132:21
 TCP TTL:18 TOS:0x0 ID:39426
 ******SF Seq: 0x5F1E103E Ack: 0x607921C9 Win: 0x404
[**] IDS198 - SCAN-SYN FIN [**]
 02/04-17:24:25.648075 211.6.213.230:53-> 210.174.164.132:53
 TCP TTL:28 TOS:0x0 ID:39426
 ******SF Seq: 0x575E1213 Ack: 0x7B8AAAC7 Win: 0x404
```

Guy Bruneau saw these same type of packets from a different source address and port numbers on September 11th and 13th of 2000 at http://www.sans.org/y2k/091500.htm

```
asctcpdump -s 1518 -x ip and host 216.242.88.30 -n -v -r tcp.2000091104
            04:33:13.713010 216.242.88.30.9704 > 192.168.30.1.9704: SF
            1468237273:1468237273(0) win 1028 (ttl 31, id 39426)
            4500 0028 9a02 0000 1f06 335a d8f2 581e
            E..(......3Z..X.
            xxxx xxxx 25e8 25e8 5783 85d9 0d50 bedb .p..%.%.W....P..
            5003 0404 e810 0000 0000 0000 0000
            P.............
```

Robin Siddique was listed as a correlation by Guy and saw traffic on August 1st, 2000: http://www.sans.org/y2k/080300.htm

```
08/01-00:04:50.529320 207.0.62.254:1524 -> MY.NET.1.3:1524
                  TCP TTL:31 TOS:0x0 ID:39426
                  **SF**** Seq: 0x59BA1B2 Ack: 0x5611503C Win: 0x404
                  00 00 00 00 00 00 ......
```

7. Evidence of active targeting:

Based upon the correlations on incidents.org, this does not appear to be targeted. It appears that many hosts are likely being scanned from this particular attacker.

8. Severity:

Severity = (Criticality [1] + Lethality [1]) - (System [3] + Network [3] Countermeasures) = -4

Criticality - 1 - This is a personal PC that I would not like to lose, but there is nothing overly important that couldn't be replaced.
Lethality - 1 - This query is mainly an attempt to query for listening ports and possibly circumvent some security equipment and the services in question are not being run. Little danger perceived.
Countermeasures
System - 3 - The system is a relatively well patched PC no running either of the services targeted.
Network - 3 - The firewall router is limiting access to other machines via NAT.

9. Defensive recommendation:

The defenses appear to be working fine in this case. The tcpdump logs were checked and only RESET responses went back. More restrictive firewall rules could be implemented to avoid any response to invalid service requests. Running minimal network and host services is the best way to minimize the risk of this attack.

10. Multiple choice test question:

What is a characteristic of a packet created by the SynScan tool?
a) All TCP flags are set
b) source and destination ports are equal
c) TCP SYN, FIN, and URGENT flags both set
d) Fragment length equal to destination port number

Answer: b

Detect #5

```
May 23 01:02:28 denied tcp 211.224.92.154(3129) -> xxx.yyy.zzz.161(139), 1 packet
May 23 01:03:49 denied tcp 211.224.92.154(3228) -> xxx.yyy.zzz.232(27374), 1 packet
May 23 01:04:02 denied tcp 211.224.92.154(3228) -> xxx.yyy.zzz.232(27374), 2 packets
May 23 01:04:05 denied tcp 211.224.92.154(3276) -> xxx.yyy.zzz.232(139), 2 packets
May 23 01:07:31 denied tcp 211.224.92.154(4754) -> xxx.yyy.zzz.196(139), 2 packets
May 23 01:07:32 denied tcp 211.224.92.154(4715) -> xxx.yyy.zzz.196(27374), 2 packets
May 23 01:15:39 denied tcp 211.224.92.154(4619) -> xxx.yyy.zzz.134(139), 2 packets
May 23 01:15:41 denied tcp 211.224.92.154(4618) -> xxx.yyy.zzz.134(12345), 2 packets
May 23 01:24:18 denied tcp 211.224.92.154(1728) -> xxx.yyy.zzz.120(27374), 1 packet
May 23 01:24:26 denied tcp 211.224.92.154(1731) -> xxx.yyy.zzz.120(139), 1 packet
May 23 01:24:27 denied tcp 211.224.92.154(1729) -> xxx.yyy.zzz.120(12345), 1 packet
May 23 02:24:19 denied tcp 211.224.92.154(2073) -> xxx.yyy.zzz.110(139), 2 packets
```

May 23 02:26:54 denied tcp 211.224.92.154(2178) -> xxx.yyy.zzz.33(27374), 1 packet
May 23 02:37:36 denied tcp 211.224.92.154(2226) -> xxx.yyy.zzz.192(139), 1 packet
May 23 02:37:37 denied tcp 211.224.92.154(2225) -> xxx.yyy.zzz.192(12345), 1 packet
May 23 02:42:47 denied tcp 211.224.92.154(1187) -> xxx.yyy.zzz.249(27374), 1 packet
May 23 02:50:00 denied tcp 211.224.92.154(3993) -> xxx.yyy.zzz.91(12345), 1 packet
May 23 02:50:06 denied tcp 211.224.92.154(3993) -> xxx.yyy.zzz.91(12345), 1 packet
May 23 03:21:42 denied tcp 211.224.92.154(2096) -> xxx.yyy.zzz.103(139), 2 packets
May 23 03:44:34 denied tcp 211.224.92.154(1633) -> xxx.yyy.zzz.253(12345), 2 packets
May 23 03:44:40 denied tcp 211.224.92.154(1632) -> xxx.yyy.zzz.253(27374), 2 packets
May 23 03:44:42 denied tcp 211.224.92.154(1643) -> xxx.yyy.zzz.253(139), 2 packets
May 23 04:04:28 denied tcp 211.224.92.154(4262) -> xxx.yyy.zzz.49(12345), 1 packet
May 23 04:09:38 denied tcp 211.224.92.154(2351) -> xxx.yyy.zzz.133(12345), 2 packets
May 23 04:32:34 denied tcp 211.224.92.154(4232) -> xxx.yyy.zzz.220(27374), 1 packet
May 23 04:32:41 denied tcp 211.224.92.154(4232) -> xxx.yyy.zzz.220(27374), 1 packet
May 23 04:32:42 denied tcp 211.224.92.154(4243) -> xxx.yyy.zzz.220(139), 1 packet
May 23 04:43:05 denied tcp 211.224.92.154(1116) -> xxx.yyy.zzz.19(27374), 2 packets
May 23 05:45:12 denied tcp 211.224.92.154(3573) -> xxx.yyy.zzz.215(139), 2 packets
May 23 05:52:32 denied tcp 211.224.92.154(2346) -> xxx.yyy.zzz.31(12345), 1 packet
May 23 06:00:46 denied tcp 211.224.92.154(3767) -> xxx.yyy.zzz.54(27374), 1 packet
May 23 06:01:03 denied tcp 211.224.92.154(3768) -> xxx.yyy.zzz.54(12345), 2 packets
May 23 06:04:59 denied tcp 211.224.92.154(4883) -> xxx.yyy.zzz.137(12345), 1 packet
May 23 06:08:25 denied tcp 211.224.92.154(1496) -> xxx.yyy.zzz.153(139), 1 packet
May 23 06:09:48 denied tcp 211.224.92.154(2235) -> xxx.yyy.zzz.53(139), 2 packets
May 23 06:11:13 denied tcp 211.224.92.154(3525) -> xxx.yyy.zzz.222(12345), 1 packet
May 23 06:11:14 denied tcp 211.224.92.154(3526) -> xxx.yyy.zzz.222(139), 1 packet
May 23 06:11:15 denied tcp 211.224.92.154(3525) -> xxx.yyy.zzz.222(12345), 1 packet
May 23 07:13:05 denied tcp 211.224.92.154(4774) -> xxx.yyy.zzz.16(27374), 1 packet
May 23 07:13:22 denied tcp 211.224.92.154(4775) -> xxx.yyy.zzz.16(12345), 2 packets
May 23 07:13:23 denied tcp 211.224.92.154(4780) -> xxx.yyy.zzz.16(139), 2 packets
May 23 08:09:05 denied tcp 211.224.92.154(3488) -> xxx.yyy.zzz.190(27374), 1 packet
May 23 08:10:41 denied tcp 211.224.92.154(3489) -> xxx.yyy.zzz.190(12345), 2 packets
May 23 08:10:46 denied tcp 211.224.92.154(3490) -> xxx.yyy.zzz.190(139), 2 packets
May 23 08:10:52 denied tcp 211.224.92.154(3488) -> xxx.yyy.zzz.190(27374), 2 packets
May 23 08:12:23 denied tcp 211.224.92.154(3979) -> xxx.yyy.zzz.131(12345), 1 packet
May 23 08:12:30 denied tcp 211.224.92.154(3981) -> xxx.yyy.zzz.131(139), 1 packet
May 23 08:12:44 denied tcp 211.224.92.154(3978) -> xxx.yyy.zzz.131(27374), 1 packet
May 23 08:12:58 denied tcp 211.224.92.154(4890) -> xxx.yyy.zzz.218(139), 2 packets
May 23 08:13:02 denied tcp 211.224.92.154(4889) -> xxx.yyy.zzz.218(12345), 2 packets
May 23 11:47:26 denied tcp 211.224.92.154(4978) -> xxx.yyy.zzz.168(12345), 1 packet
May 23 12:06:19 denied tcp 211.224.92.154(3225) -> xxx.yyy.zzz.188(27374), 1 packet
May 23 12:16:26 denied tcp 211.224.92.154(3321) -> xxx.yyy.zzz.117(139), 2 packets
May 23 13:25:23 denied tcp 211.224.92.154(2936) -> xxx.yyy.zzz.26(12345), 1 packet
May 23 13:26:29 denied tcp 211.224.92.154(1259) -> xxx.yyy.zzz.149(139), 1 packet
May 23 13:30:48 denied tcp 211.224.92.154(1281) -> xxx.yyy.zzz.215(12345), 2 packets
May 23 13:40:42 denied tcp 211.224.92.154(3824) -> xxx.yyy.zzz.81(139), 2 packets
May 23 13:58:38 denied tcp 211.224.92.154(4917) -> xxx.yyy.zzz.89(12345), 2 packets
May 23 13:58:40 denied tcp 211.224.92.154(4916) -> xxx.yyy.zzz.89(27374), 2 packets
May 23 14:16:26 denied tcp 211.224.92.154(2897) -> xxx.yyy.zzz.99(12345), 2 packets
May 23 14:16:28 denied tcp 211.224.92.154(2900) -> xxx.yyy.zzz.99(139), 2 packets
May 23 14:22:32 denied tcp 211.224.92.154(2152) -> xxx.yyy.zzz.90(27374), 2 packets
May 23 14:22:34 denied tcp 211.224.92.154(2158) -> xxx.yyy.zzz.90(139), 2 packets
May 23 14:22:35 denied tcp 211.224.92.154(2153) -> xxx.yyy.zzz.90(12345), 2 packets
May 23 14:45:28 denied tcp 211.224.92.154(1064) -> xxx.yyy.zzz.79(27374), 1 packet
May 23 14:45:50 denied tcp 211.224.92.154(1064) -> xxx.yyy.zzz.79(27374), 2 packets
May 23 14:47:28 denied tcp 211.224.92.154(3398) -> xxx.yyy.zzz.8(12345), 2 packets
May 23 15:11:33 denied tcp 211.224.92.154(4620) -> xxx.yyy.zzz.68(27374), 1 packet
May 23 15:11:51 denied tcp 211.224.92.154(4622) -> xxx.yyy.zzz.68(139), 2 packets
May 23 15:11:52 denied tcp 211.224.92.154(4620) -> xxx.yyy.zzz.68(27374), 2 packets

1. Source of Trace:

This trace above was taken from Ken Connelly's post of May 24, 2001 to incidents.org:
http://www.incidents.org/archives/intrusions/msg00444.html

```
% Rights restricted by copyright. See http://www.apnic.net/db/dbcopyright.html
 inetnum:     211.217.0.0 - 211.225.255.255
  netname:     KORNET
  descr:       KOREA TELECOM
  descr:       KOREA TELECOM Internet Operating Center
  country:     KR
  admin-c:     GP33-AP
  tech-c:      WK44-AP
  mnt-by:      MNT-KRNIC-AP
  changed:     hostmaster@apnic.net 20000912
  source:      APNIC
```

2. Detect was generated by:

Ken notes in his post that this trace is from a Cisco ACL log, so this is likely from a border router. This log is fairly straightforward and minimal. It lists on each line the

date, time, whether the packet was permitted or denied, protocol (tcp), source IP address (211.224.92.154), source port number within parenthesis, an arrow indicating that the source was on the left and the destination on the right, destination IP address (xxx.yyy.zzz.nnn), destination port number (12345, 27374, and 139), and finally the number of packets meeting these characteristics. The times in the log are in the Eastern Standard Time Zone (GMT -5).

3. Probability the source address was spoofed:

The odds are that the source address was not spoofed. We do not know the TCP flags associated with each packet in the log. These ports suggest that there is an attempt to check for the presence of a process responding on the ports probed and a legitimate address must be used to receive a response. Due to the corroborated traffic shown in section 6 and the reputation of the Korean addresses, it seems very probable that these are valid addresses.

4. Description of the attack:

On May 23, 2001, over 14 hours between 1:02 am and 3:11 pm, a series of TCP packets are being sent from a single IP address to many hosts to three separate port numbers: 139, 27374, and 12345. The packets collectively addressed 39 of the 254 host addresses within this class C address space between xxx.yyy.zzz.8 and xxx.yyy.zzz.253. Each address is probed is not necessarily checked for all 3 of the ports and the ports that are checked are done in random order. All of the ports that are directed to a given address are grouped together before another address is tested. No more than 3 packets were sent to a particular combination of address and port. On average the time between targets in the log being hit was 21 minutes and 36 seconds. There was a three and a half hour quiet gap from 8:13 when xxx.yyy.zzz.218 was probed until 11:47 when xxx.yyy.zzz.168 was the next target. On average over all of the packets, there were 6 seconds during which packets were sent to a particular target address with a maximum of 1 minute and 47 seconds for xxx.yyy.zzz.190 in which all three ports were attempted and retried. Because of these times and the corroborations that others are being hit by this attacker, it is presumed that during the average 21 minute gap somebody else is being visited at probably this same set of ports.

Common Vulnerabilities and Exposures (CVE):

Name	Description
CVE-1999-0153	Windows 95/NT out of band (OOB) data denial of service through NETBIOS port, aka WinNuke.
CAN-1999-0660	A hacker utility or Trojan Horse is installed on a system, e.g. NetBus, Back Orifice, Rootkit, etc.
CAN-2000-0138	A system has a distributed denial of service (DDOS) attack master, agent, or zombie installed.

5. Attack mechanism:

All of these attacks are fairly specific to PCs, although there may be some vulnerability on port 139 of a Samba server that could be exploited. Packets are delivered to potential victim machines at popular default ports for common Trojans in the hopes of receiving a response. If the attacker intended to use WinNuke, the packet would have the urgent flag set and if the machine was not patched, then the blue screen of death would result. Assuming that the packets being sent are SYN packets, then a SYN ACK response would be interpreted to mean that the probed port was active and could potentially be exploited. Netbus and SubSeven trojans provide remote control of the machine they are running on. The testing of the ports for a given address implies that one program is being used for the scanning to provide the coordination and there might be some sort of random test to see which port to test next to account for there being no apparent order.

Destination Port Number	Number of Packets	Port Description
139	40	Winnuke, netbios session
12345	36	GabanBus, Netbus 1.x, Pie Bill Gates, X - Bill, My Pics, Whack Job, Ultor
27374	31	SubSeven

6. Correlations:

incidents.org shows 15 similar TCP packets between May 17th - 23rd addressed for ports 139, 27374, 12345 at:
http://www.incidents.org/cid/search_result.php?source=211.224.92.154&minsport=0&maxsport=65536&mintport=0&maxtport=65536&minproto=0&maxproto=99&mindate=0000-00-00&maxdate=2002-01-01&flags=*&Submit=Submit

dshield.org shows 121 records for packets emanating from this IP address covering the time from December 23, 2000 until May 27, 2001 against 22 different targets. The same 3 ports are shown as in our traffic: 139, 27374, 12345.
http://www1.dshield.org/ipinfo.php?ip=211.224.92.154

7. Evidence of active targeting:

There is no reason to believe that this network is being singled out, but they are making a concerted effort to find a potential box to compromise. We are not sure whether this network is PC centric which would raise concerns that they may have reason to come after this network in particular. Because of the general probing, I suspect this falls under reconnaissance.

8. Severity

Severity = (Criticality [3] + Lethality [5]) - (System [4] + Network [4] Countermeasures) = 0

Criticality - 3 - There could be Window-based servers within this network, we don't know.
Lethality - 5 - The attacker would likely be able to control any box that responded to this probe.
Countermeasures
System - 4 - We have no information about the systems being protected, but we will assume that they are patched well.
Network - 4 - We have litle information about the network, except that the router was configured well enough to rebuff such attacks.

9. Defensive recommendation:

The defenses are holding for now. Some things to consider if they are not in place would be a general firewall behind the router with a restrictive ruleset (only permit what

is absolutely necessary). Ensure that Netbios ports are prevented from crossing the router and firewall. Ensure that virus protection is running on personal computers with updated signatures. Consider personal firewalls and ensure only necessary network services are running on each host.

10. Multiple choice question:

Cisco ACL logs do NOT contain:
a) Date and time of packet
b) Number of packets
c) TCP flags
d) Source port number

Answer: c

Assignment 2 - Describe the State of Intrusion Detection - IDS technology

Basic Security Module: Solaris Kernel-based Auditing

By Donald Pitts
SANS IDS GCIA Practical 2.8
May 23, 2001

Purpose

This paper will provide an introduction to the concepts involved with the SunSHIELD Basic Security Module (BSM).

Introduction

Kernel-based auditing on the Solaris platform is available with the activation of BSM which is included within the standard Operating System (OS). [1-3] Many commercial host-based Intrusion Detection Systems (IDS) running on Solaris machines utilize BSM features internally. An intrusion detection analyst can gain powerful insight into their enterprise by understanding and utilizing this auditing system either directly or through a third party package such as Emerald eXpert-BSM, ISS RealSecure, or PGP Security CyberCop Monitor. [4-6] The existence of the Linux BSM project, seeking to provide a similar capability for Linux, is an indication of the viability and usefulness of BSM. [7]

Background

BSM was originally developed to add the capabilities otherwise lacking in standard UNIX which were necessary to meet the C2 level of security as outlined in the Trusted Computer System Evaluation Criteria (TCSEC) "Orange" book. [8-9] In practice today, the Common Criteria for Information Technology Security Evaluation (CC) has largely supplanted the TCSEC. [10-11] While this paper focuses on auditing, other features are also added, of which administrators should be aware. The Stop-a key combination, which drops directly into the Boot PROM, is disabled. Restrictions are also added for removable media to limit access (both read and write) to a single user while the media is loaded in the machine through an allocation/deallocation scheme. [12] Sun does not provide a mechanism to easily enable only the auditing aspects of BSM. Listing 11 within Appendix A has a patch you can apply to your copy of BSM to only enable the auditing features.

Enabling BSM

The necessary files are installed with the core installation, but must be activated by the administrator by entering single-user mode (**init 1**), activating BSM (**/etc/security/bsmconv**), and then rebooting into multi-user mode with BSM activated (**init 6**). Should BSM subsequently need to be disabled, the same procedure can be followed with the substitution of **/etc/security/bsmunconv**.

Who was that anyway?

BSM tracks each user based upon the account used to first successfully access the system despite any subsequent host-based changes of identity such as with the **su** command. This initial user identity is known as the Audit User ID (UID). Therefore a distinct user account should be constructed for each user of the system which should be used before assuming any administrator capabilities. The changes in identity are not tracked across a transition made using telnet and similar networked access.

What just happened?

An audit event corresponds to a system call, particular usage of a given system call, or some significant action of note. The standard BSM delivers some user-level audit events such as those dealing with **login** and **ftp**. [12] Figuring out what happened from a BSM auditing is very similar to looking at output from the **truss** command. An understanding of the system at a lower level is required to interpret BSM output compared to that required to decipher most syslog messages which are at the application level. Often a user will execute a command which will in turn cause tens or even hundreds of system calls to be made. This is both powerful in terms of precise visibility into the innerworkings of the system as well as unwieldy with regard to the potential volume of information to be examined. Another consideration is that enough events of interest must be configured to be captured so that the big picture can be reasonably reconstructed with some assurance.

Audit events are grouped into audit classes to simplify the configuration process for the administrator. Note that a given audit event can be in more than one audit class and will be audited if one or more of the relevant classes are selected. BSM ships with a predefined set of audit classes shown in Table 1 below.

Flag	Description	Representative Events
no	Special class indicating nothing	
fr	File read	open(2) with read flag specified, readlink(2)
fw	File write	open(2) with write flag specified
fa	File attribute access	access(2), stat(2), statfs(2)

fm	File attribute modification	chmod(2), chown(2), fcntl(2), acl(2)
fc	File creation	creat(2), link(2), rename(2),mkdir(2), open(2) with creat flag specificed, process dumped core
fd	File deletion	unlink(2), rename(2), truncate(2), open(2) with trunc flag specified
cl	File closure	close(2), munmap(2)
pc	Process	exit(2), fork(2), exec(2), chdir(2), execve(2), kill(2), chroot(2)
nt	Network events	connect(2), accept(2), bind(2), socket(2), sendto(2), sendmsg(2), putmsg(2)
ip	System V Inter-Process Communications (IPC)	shmget(2), msgget(2), semget(2), semctl(2), shmctl(2)
na	Events not attributable to a particular user	system booted, enter prom, exit prom, inetd, mountd mount
ad	Administrative actions	mknod(2), reboot(2), settimeofday(2), unmount(2), exportfs(2), auditon(2), modctl(2), halt(1m), shutdown(1b), create user
lo	Login/logout	logins - local, telnet, rlogin, rsh access, su, rexecd, passwd, rexd, ftp access
ap	Application	none
io	Ioctl	ioctl(2)
ex	Executing programs	exec(2), execve(2)
ot	Other	memcntl(2)
all	Special hierarchical class indicating all classes	

Table 1 - Predefined Audit Classes

Event and Class Customization

The classes and the assignment of events to classes comes predefined, but can be modified. Events can be added and class membership for existing events can be altered or expanded. A specific set of customizations to audit classes for BSM can be found in the Sun BluePrints article "Auditing in the Solaris 8 Operating Environment". [2-3] Similar to syslog, third party programs can incorporate special library calls to provide application-level BSM logging.

The audit_class file contains the current set of audit classes shown in Listing 1 below. The audit_event file defines the events and their mapping to classes shown in Listing 2 below. A complete list of the audit events within a particular audit class, substituted for *flag*, can be shown for your machine with the command:

cat /etc/security/audit_event | nawk -F: '$4 ~ "*flag*" {print $2 " " $3}'

```
...
# User Level Class Masks
#
...
# File Format:
#
# mask:name:description
#
0x00000000:no:invalid class
0x00000001:fr:file read
...
```

Listing 1 - Portion of /etc/security/audit_class file

```
...
# Audit Event Database
#
# File Format:
#
# event number:event name:event description:event classes (comma separated)
#
...
# kernel audit events
#
0:AUE_NULL:indir system call:no
1:AUE_EXIT:exit(2):pc
...
7:AUE_EXEC:exec(2):pc,ex
...
```

Listing 2 - Portion of /etc/security/audit_event file

Tuning

Why not just turn on all auditing for the entire system and be done with it? As you might expect, a large amount of data can be generated very quickly - even if you limit yourself to only failures. So the quantity of disk storage can be a problem as well as the added time overhead to do the logging. Appropriate tuning according to your needs

is the most critical aspect of successfully deploying BSM. Performance will be affected roughly 5-10%, but can be improved by limiting auditing to necessary audit events. [13-14]

When beginning it would be advisable to limit auditing to a few specific users or a very limited set of audit classes while becoming familiar with the ramifications to your environment. Investigating BSM in a lab environment ahead of time would also be recommended while you are learning.

Controlling What is Audited

Selecting what to audit is done at the audit class level. Each audit class has an abbreviated name called an audit flag that is typically two characters (i.e. fr). A prefix character of a minus (-) or plus (+) can be added to limit auditing to failures (i.e. -fr) or successes (i.e. +fr) respectively. The absence of a prefix character (i.e. fr) indicates that these events are to be audited regardless of the outcome. Multiple classes can be listed by separating them with commas and no spaces (i.e. ad,-lo).

What if you needed to specify that all failed events except those in the file_read (fr) class should be audited? The only way to do this with what you have learned so far is to list explicitly all of the other classes each with a minus sign prefix:

-fw,-fa,-fm,-fc,-fd,-cl,-pc,-nt,-ip,-na,-ad,-lo,-ap,-io,-ex,-ot

While this works, it is unwieldy and suffers in flexibility when new audit classes are defined. As an alternative, an additional prefix of a caret (^) can be placed at the very beginning of an audit flag and any other associated prefix. This specifies that a previously specified audit flag be treated as though it was never indicated. Thus, the improved approach would thus look like:

-all,^-fr

Table 2 summarizes the different audit prefixes which are supported.

Prefix Usage	Description
flag	Everything - successes and failures (no prefix)
+*flag*	Limited to successes
-*flag*	Limited to failures
^*flag*	Negates earlier presence of successes and/or failures
^+*flag*	Negates earlier presence of successes
^-*flag*	Negates earlier presence of failures

Table 2 - Audit Prefix Combinations

Scope of Control

Flags can be used in three distinct cases: system-wide, user specific, or when the responsible user cannot be determined. Examples when an event is not attributable to a user is when the machine is powered up or an invalid username is provided at login. Each of these scopes can be controlled within specific configuration files as indicated in Table 3. An example of an audit_control file is shown in Listing 3 where the system-wide and non-attributable audit classes can be specified. If any changes are made to the audit_control file, then the command **audit -s** can be issued so that the audit daemon reloads its configuration. Any changes will only affect new logins and not any existing login sessions.

Auditing Scope	Audit Flags Location
Non-attributable	"naflags:" line in /etc/security/audit_control
System-wide	"flags:" line in /etc/security/audit_control
User specific	User specific line in /etc/security/audit_user

Table 3 - Audit Class Selection Scope

```
dir:/var/audit
flags:lo
minfree:20
naflags:lo,ad
```

Listing 3 - Portion of example /etc/security/audit_control file

Singling out a User

Each line within the audit_user (refer to Listing 4) file consists of three columns separated by colons. The first column is the username. The second column is known as the "always audit" field. Any flags specified here complement those already specified system-wide. The third column is the "never audit" field and will override the corresponding flag should it also appear within the "always audit" field for the user or even if designated system-wide.

```
...
# User Level Audit User File
#
# File Format
#
# username:always:never
#
```

```
root:lo:no
audtest:ex,-all:no
```

Listing 4 - Portion of example /etc/security/audit_user file

Audit Trail Files

The files which contain the actual audit records are called audit trail files. The data is stored in binary form. The audit daemon knows which directories can be used for storage of these files by examining the "dir:" entries of the audit_control file (shown in Listing 3 previously). By default, audit trail files are stored within /var/audit. Listing 5 shows some example audit trail filenames for a machine named *generic*. These files are named with the starting and ending timestamps. While an audit trail file is still in use, the designation "not_terminated" is utilized within the filename where the ending timestamp will eventually be located. The name of the machine generating the file is also added as a suffix to the filename.

Audit trail files should periodically be closed and new ones created to prevent them from getting too large. This can be achieved based upon the maximum size of an audit file (**auditconfig -setfsize *n*** within /etc/security/audit_startup) or chronologically by placing the **audit -n** command within cron. [15]

Audit trail files are not deleted by BSM. An automated or manual procedure will be needed to potentially archive and remove old audit trail files. By default audit records are dropped when all allocated disk and memory buffers are full. This behaviour can be modified by removing **auditconfig -setpolicy +cnt** from /etc/security/audit_startup.

```
20010507201838.20010507202606.generic
20010507202606.20010507221213.generic
20010507221213.20010507221652.generic
20010507221652.20010507221810.generic
20010507221810.20010507222332.generic
20010507222332.20010507223056.generic
20010507223056.20010507223151.generic
20010507223151.20010507223624.generic
20010522214103.not_terminated.generic
```

Listing 5 - Example of Audit Trail Filenames

Audit Records and Tokens

Each time an event to be audited occurs, an audit record is logged in the audit trail. Each audit record consists of a set of one or more tokens. Each token represents a related set of information of interest to auditing. A given token will appear in multiple records because they are general in nature and applicable to many different audit events. Each audit event has a set of tokens which are relevant and may be included in a particular occurrence based upon the context of the situation.

A header token is the first token in each audit record identifying the type of event and the date/time, except for the file token which encompasses its own special audit record. The file token appears only at the beginning and the end of a physical audit trail file documenting when the file was initiated and completed. A subject token shows who caused the event with various UIDs, group membership, process ID, session ID, and a terminal identifier. The inclusion of some tokens are optional and configurable by the administrator. Ordinarily, the end of an audit record can be identified by the presence of a header or file token relating to a subsequent audit record. A trailer token can be activated to explicitly mark the end of the audit record by **auditconfig -setpolicy +trail**. The presence of a trailer token can confirm an audit record was logged in its entirety. This token can also be helpful to flush a buffered audit record into being processed immediately by a tool that might otherwise wait until another audit record is logged. Flushing would only be an issue when very few events are being logged and a significant delay might exist between events. The **auditconfig -setpolicy +argv** command will cause the arguments corresponding to commands executed with an exec system call to be logged in an exec_args token**.** Auditconfig statements are typically added to /etc/security/audit_startup so the same policy will still be enforced when a machine reboots.

Table 4 shows a summary of the audit tokens constructed from the "Audit Record Format" (Chapter 3) and "Audit Token Structure" (Appendix A) sections of the SunSHIELD Basic Security Module Guide. [1]

Token Type	Description
acl	Indicates the type, ID, and permissions for an Access Control List (ACL).
arbitrary	Generalized way to include an array not provided for in other tokens.
arg	Holds the value for one argument and its position within the system call. Multiple of these can be present within an audit record.
attr	Contains owner, group, file system, inode, and permissions from the vnode for a file.
exec_args	Optionally lists the parameters passed to an exec system call. Requires **auditconfig -setpolicy +argv** to activate.
exec_env	Optionally specifies the environment variables during an exec system call. Requires **auditconfig -setpolicy +arge** to activate.
exit	Status returned from a program as it ended.
file	Designates the beginning and ending of a physical audit trail file indicating the date, time, and filename.
groups	Obsolete - see newgroups token.
header	Begins every audit record except the special file audit records. Contains time and type of event this record represents.
in_addr	Specifies an Internet Protocol (IP) address.
ip	Lists the first 20 bytes of the IP header.
ipc	Indicates the handle for a specific IPC object (message, semaphore, or shared memory).
ipc_perm	Holds the IPC owner, creator, permissions, sequence number, and key value.
iport	Contains an TCP or UDP port number.
newgroups	Optionally lists the groups a process is a member of. Requires **auditconfig -setpolicy +group** to activate.
opaque	Stores unformatted data as bytes.

path	The full filename for an object is provided.
process	For a process being acted upon, this token identifies the user running the process with audit, real, and effective IDs as well as the associated PID, session ID, and terminal ID.
return	Records a status message and return value from a system call.
seq	Optionally holds a unique identifying number for each audit record. Requires **auditconfig -setpolicy +seq** to activate.
socket	Describes an Internet socket with the IP addresses, ports, and type (TCP, UDP, or UNIX).
socket-inet	Holds information for a socket connection to a local port including family (AP_INET, etc.), port number, and socket address.
subject	For a process performing an audited action, this token identifies who performed the action with audit, real, and effective IDs as well as PID, session ID, and terminal ID.
text	Contains a single text string.
trailer	Optionally ends each audit record except for the special file audit records. Requires **auditconfig -setpolicy +trail** to activate.

Table 4 - Audit Tokens

Viewing Audit Trails

Once auditing is enabled and operational the audit trail may be examined. The two primary tools provided by Sun for this purpose are **auditreduce** and **praudit** - neither of which provide a GUI. The **auditreduce** command can collate events from multiple audit trail files (potentially from multiple machines) and permit filtering based upon time range, user, group, process ID, audit class, event, machine, and object type. The output of **auditreduce** as well as a raw audit trail file are encoded as binary and must be converted to ASCII with **praudit** before viewing. So the simplest command is to show the entire audit trail (**auditreduce | praudit**). An example of the output from **praudit** is in Listing 6. Each token is on a separate line with a keyword for the token followed by the values for the fields. The fields are not explicitly marked as to their identity. A knowledge of the fields and their order for each of the tokens is necessary to be able to properly interpret this output. GUI-based tools from sources other than Sun are available to view audit trail files. [14,16-17] A text-based tool is also available to display audit trails in an annotated format. [18]

```
header,168,2,execve(2),,Fri May 18 17:33:23 2001, + 223716593 msec
path,/usr/bin/cp
attribute,100555,bin,bin,32,125552,0
exec_args,3,
cp,/etc/shadow,my_copy_of_shadow
subject,audtest,audtest,staff,audtest,staff,9137,9129,24 9 localhost
return,success,0
trailer,168
header,122,2,stat(2),,Fri May 18 17:33:23 2001, + 233718216 msec
path,/export/home/audtest/my_copy_of_shadow
subject,audtest,audtest,staff,audtest,staff,9137,9129,24 9 localhost
return,failure: No such file or directory,-1
trailer,122
header,130,2,open(2) - read,,Fri May 18 17:33:23 2001, + 233718216 msec
path,/etc/shadow
attribute,100400,root,sys,32,441096,0
subject,audtest,audtest,staff,audtest,staff,9137,9129,24 9 localhost
return,failure: Permission denied,-1
trailer,130
header,174,2,ioctl(2),,Fri May 18 17:33:37 2001, + 813785319 msec
path,/etc/security/audit_control
attribute,100640,root,sys,32,902278,0
argument,2,0x5401,cmd
argument,3,0xffbef2b4,arg
subject,audtest,root,root,root,root,9129,9129,24 9 localhost
return,failure: Inappropriate ioctl for device,-1
trailer,174
header,171,2,ioctl(2),,Fri May 18 17:33:37 2001, + 813785319 msec
path,/etc/security/audit_user
attribute,100640,root,sys,32,902280,0
argument,2,0x5401,cmd
argument,3,0xffbef2bc,arg
subject,audtest,root,root,root,root,9129,9129,24 9 localhost
return,failure: Inappropriate ioctl for device,-1
trailer,171
header,172,2,ioctl(2),,Fri May 18 17:33:37 2001, + 813785319 msec
path,/etc/security/audit_event
attribute,100644,root,sys,32,902277,0
argument,2,0x5401,cmd
argument,3,0xffbef4bc,arg
subject,audtest,root,root,root,root,9129,9129,24 9 localhost
return,failure: Inappropriate ioctl for device,-1
trailer,172
header,89,2,login - telnet,,Fri May 18 17:47:24 2001, + 518069161 msec
subject,-1,-1,-1,-1,-1,9163,9163,24 9 localhost
text,invalid user name
return,failure: No such process,-1
trailer,89
```

Listing 6 - Output from praudit Command

Having Trouble?

More than once you may wonder why you are not seeing an audit record you are expecting. Most often the problem is that the audit event is not configured properly to be captured. This can be for the particular outcome (success or failure) or at a particular scope (global, user, or non-attributable).

Road Map

A fairly extensive listing of the directories, configurable data files, and program executables relating to BSM auditing is provided in Table 5.

File or Directory	Description
/etc/security/audit_class	Defines classes valid on the system.
/etc/security/audit_control	Defines directories to store audit trail files ("dir:"), classes to globally audit ("flags:"), percentage threshold of excess storage within the audit trail directories below which warnings are generated ("minfree:"), and the classes to audit for non-attributable events ("naflags:").
/etc/security/audit_data	Contains the Process ID (PID) of the current audit daemon and the full filename of the current audit trail file where audit records are being actively stored.
/etc/security/audit_event	Defines events valid on the system and their class membership.
/etc/security/audit_startup	Configurable shell script created when BSM is enabled and contains auditconfig statements defining the local auditing policy which are executed by /etc/init.d/audit upon boot of the machine.
/etc/security/audit_user	Defines classes to audit or not audit for each user which augment/override those defined globally in the audit_control file.
/etc/security/audit_warn	Shell script run by audit daemon to warn administrator of any audit difficulties.
/etc/security/bsmconv	Shell script run in single-user mode by administrator to enable BSM
/etc/security/bsmunconv	Shell script run in single-user mode by administrator to disable BSM
/var/audit	Default directory for storage of audit trail files
/etc/init.d/audit	Initialization and termination script for BSM auditing.
/usr/sbin/auditd	Audit daemon binary executable.
/usr/sbin/audit	Command to control the auditing daemon. Options include starting a new audit trail file, rereading the audit control file, and terminating the audting daemon.
/usr/sbin/auditconfig	Command to configure, examine, or check auditing parameters. Typically used in /etc/security/audit_startup.
/usr/sbin/auditreduce	Command to combine and filter audit data from one or more audit trail files. Result is still in binary form.
/usr/sbin/auditstat	Command to display auditing statistics
/usr/sbin/praudit	Command to convert binary audit trail data into human readable ASCII format.

Table 5 - Audit Relevant Commands, Configuration Files, and Directories

Using BSM for Host-based IDS

As a proof of concept, a shell script, **bsmmon**, was developed that processes the current audit trail file from BSM looking for any events of interest. The listing of the script is included as Listing 12 in Appendix B. Each event is examined independently of the others and no attempt is made to extend this examination across multiple audit events.

An example telnet session using an account being audited is shown in Listing 7 and some of the corresponding raw audit trail records are in the previously referenced Listing 6. The session was logged from the execve(2) audit records and is shown in Listing 8. The initial four statements in the session correspond to activities performed by the system on behalf of the user prior to providing the shell prompt. This is an example where investigation can be required to differentiate between what the user explicitly is doing and what is a side effect produced by the system.

The **bsmmon** script displays alerts on stdout by default. The only events it monitors are bad logins and access (successful or attempted) to sensitive files. An example is shown in Listing 9. The alerts can be redirected to syslog as displayed in Listing 10. Many enhancements could be made to more thoroughly examine the audit trail, but this script achieves the purpose of demonstrating what can be accomplished with the raw BSM technology. [9]

```
# telnet 0
login: audtest
Password:
Last login: Fri May 18 17:28:31 from localhost
Sun Microsystems Inc. SunOS 5.7 Generic October 1998
$ ls -l
total 6
-rw-r--r-- 1 audtest staff 124 May 8 10:54 local.cshrc
-rw-r--r-- 1 audtest staff 581 May 8 10:54 local.login
-rw-r--r-- 1 audtest staff 562 May 8 10:54 local.profile
---s--s--x 1 audtest staff 0 May 11 09:29 stuff
$ cat local.profile
...
$ ls
local.cshrc local.login local.profile stuff
$ cp /etc/shadow my_copy_of_shadow
cp: cannot open /etc/shadow: Permission denied
$ exit
Connection closed by foreign host.
```

Listing 7 - Example UNIX Shell Session

```
# cat session_audtest_9129
Fri May 18 17:32:15 2001 + 123393825 msec, /usr/bin/sh, -sh, success
Fri May 18 17:32:15 2001 + 133393137 msec, /usr/lib/fs/ufs/quota, /usr/sbin/quota, success
Fri May 18 17:32:15 2001 + 163391553 msec, /usr/bin/cat, /bin/cat -s /etc/motd, success
Fri May 18 17:32:15 2001 + 173393335 msec, /usr/bin/mail, /bin/mail -E, success
Fri May 18 17:32:26 2001 + 253445674 msec, /usr/bin/ls, ls -l, success
```

```
Fri May 18 17:32:32 2001 + 703477600 msec, /usr/bin/cat, cat local.profile, success
Fri May 18 17:32:57 2001 + 893595415 msec, /usr/bin/ls, ls, success
Fri May 18 17:33:23 2001 + 223716593 msec, /usr/bin/cp, cp /etc/shadow my_copy_of_shadow, success
```

Listing 8 - Session Actions Extracted from Audit Trail

```
**************************************************
Access to sensitive file
Time: Fri May 18 17:33:23 2001 + 233718216 msec
--------------------------------------------------
Event ID: open(2) - read
Filename: /etc/shadow
Audit ID: audtest
Status: failure
Status Message: Permission denied
File Owner: root
Return Value: -1
File System ID: 32
File Device ID: 0
Effective Group ID: staff
Session ID: 9129
Real User ID: audtest
Real Group ID: staff
Inode: 441096
Process ID: PID: 9137
File Group: sys
Terminal: 24 9 localhost
File Mode: 100400
Effective User ID: audtest
--------------------------------------------------
**************************************************
Login failure
Time: Fri May 18 17:47:24 2001 + 518069161 msec
--------------------------------------------------
Event ID: login - telnet
Audit ID: -1
Session ID: 9163
Status: failure
Status Message: No such process
Return Value: -1
Textual Comment: invalid user name
Effective Group ID: -1
Process ID: PID: 9163
Terminal: 24 9 localhost
Effective User ID: -1
--------------------------------------------------
```

Listing 9 - Alert Log

```
May 18 17:55:20 mdess09 bsmmon: Access to sensitive file  Tm: Fri May 18 17:33:23 2001 + 233718216 msec  Evnt: open(2) -
read
May 18 17:55:20 mdess09 bsmmon:      Fl: /etc/shadow  AUID: audtest  Sts: failure  StsMsg: Permission denied  Own: root
May 18 17:55:20 mdess09 bsmmon:      Rtn: -1  FSID: 32  DID: 0  EGID: staff  SID: 9129  RUID: audtest  RGID: staff
May 18 17:55:20 mdess09 bsmmon:      Inode: 441096  PID: PID: 9137  Grp: sys  Trm: 24 9 localhost  Mode: 100400
May 18 17:55:20 mdess09 bsmmon:      EUID: audtest
May 18 17:55:20 mdess09 bsmmon: Login failure  Tm: Fri May 18 17:47:24 2001 + 518069161 msec  Evnt: login - telnet
May 18 17:55:20 mdess09 bsmmon:      AUID: -1  SID: 9163  Sts: failure  StsMsg: No such process  Rtn: -1  Txt: invalid user
name
May 18 17:55:20 mdess09 bsmmon:      EGID: -1  PID: PID: 9163  Trm: 24 9 localhost  EUID: -1
```

Listing 10 - Alerts Redirected to Syslog

Appendix A

Patch to bsmconv

The **/etc/security/bsmconv** script is provided by Sun to enable BSM capabilities. This script does not provide a means to avoid implementing the non-auditing features of BSM. Therefore, a set of changes to the default Sun script are provided with Listing 11. The changes identified here cause only the auditing features of BSM to be activated. Running **sum /etc/security/bsmconv** beforehand should show "47463 9 /etc/security/bsmconv" if you have the same version (annotated as "bsmconv.sh 1.14 98/01/21") that comes with at least Solaris 7 and 8. Change to the /etc/security directory with **cd /etc/security**, save the contents of Listing 11 to a file named bsmconv.patch, and then issue the command **patch -b -c -i bsmconv.patch bsmconv**. This should implement the changes automatically to the bsmconv script or, alternatively, it should be easy enough to apply these changes manually.

```
*** bsmconv Wed Jan  5 18:25:45 2000
--- bsmconv.new Tue May 22 13:28:21 2001
***************
*** 4,9 ****
--- 4,18 ----
  #
```

```
  # Copyright (c) 1993 by Sun Microsystems, Inc.
  #
+ # ------------------------------------------- MOD Begin 5/5/2000
+ # By default, only auditing is enabled
+ # Change these settings to "N" to enable
+ # the other aspects of the Basic Security Module
+ #
+ SKIP_DEVICE_ALLOC="Y"
+ SKIP_STOP_A="Y"
+ # ------------------------------------------- MOD End 5/5/2000
+
  PROG=bsmconv
  STARTUP=/etc/security/audit_startup
  DEVALLOC=/etc/security/device_allocate
***************
*** 98,103 ****
--- 107,116 ----
  chgrp sys ${ROOT}/${STARTUP} > /dev/null 2>&1
  chmod 0744 ${ROOT}/${STARTUP} > /dev/null 2>&1

+ # ------------------------------------------- MOD Begin 5/5/2000
+ if [ "$SKIP_DEVICE_ALLOC" != "Y" ]; then
+ # ------------------------------------------- MOD End 5/5/2000
+
  # move aside volume manager init file to prevent
  # running volume manager when bsm is enabled

***************
*** 118,123 ****
--- 131,140 ----
      mv ${ROOT}/etc/rc2.d/S92volmgt ${ROOT}/etc/security/spool/S92volmgt
  fi

+ # ------------------------------------------- MOD Begin 5/5/2000
+ fi
+ # ------------------------------------------- MOD End 5/5/2000
+
  # Turn on auditing in the loadable module

  form=`gettext "%s: INFO: turning on audit module."`
***************
*** 129,135 ****
--- 146,162 ----

  grep -v "c2audit:audit_load" ${ROOT}/etc/system > /tmp/etc.system.$$
  echo "set c2audit:audit_load = 1" >> /tmp/etc.system.$$
+
+ # ------------------------------------------- MOD Begin 5/5/2000
+ if [ "$SKIP_STOP_A" != "Y" ]; then
+ # ------------------------------------------- MOD End 5/5/2000
+
  echo "set abort_enable = 0" >> /tmp/etc.system.$$
+
+ # ------------------------------------------- MOD Begin 5/5/2000
+ fi
+ # ------------------------------------------- MOD End 5/5/2000
+
  mv /tmp/etc.system.$$ ${ROOT}/etc/system
  grep "set c2audit:audit_load = 1" ${ROOT}/etc/system > /dev/null 2>&1
  if [ $? -ne 0 ]
***************
*** 140,145 ****
--- 167,176 ----
      printf "${form}\n" $PROG
  fi

+ # ------------------------------------------- MOD Begin 5/5/2000
+ if [ "$SKIP_DEVICE_ALLOC" != "Y" ]; then
+ # ------------------------------------------- MOD End 5/5/2000
+
  # Initial device allocation files

  form=`gettext "%s: INFO: initializing device allocation files."`
***************
*** 153,158 ****
--- 184,193 ----
   mkdevmaps > ${ROOT}/$DEVMAPS
  fi

+ # ------------------------------------------- MOD Begin 5/5/2000
+ fi
```

```
+ # ---------------------------------------------- MOD End 5/5/2000
+
  }

  # main loop
```

Listing 11 - Patch for /etc/security/bsmconv

Appendix B

BSM Monitor (bsmmon) script

```
#!/bin/ksh

# Program: BSM Monitor (bsmmon)

# Author: Donald Pitts

# Date: May 18, 2001

# Purpose: Proof of concept code to show that audit events collected
#          from the Solaris Basic Security Module (BSM) could be
#          examined and filtered on-the-fly to determine whether to
#          log or act on it in some fashion.  This is designed to
#          always be running while auditing is active.

#          Three specific things are examined in the processed audit
#          records:
#          1) Access or attempted access to any sensitive files.
#             The current list is just representative and would
#             need to be improved significantly.  Each item in the
#             sensitive file list is treated as a regular expression.
#          2) Any bad logins are logged.  This could be enhanced to
#             keep a running tab based upon id and/or the source
#             and possibly dynamically adjust auditing or log
#             in a more dramatic fashion.
#          3) A file is created for each distinct Session ID indicated
#             by BSM and anything run with the execve(2) system call is
#             put in this session file.  This is meant to be able
#             to provide a record of what was likely done directly
#             by the human within the UNIX shell.

# Audit Classes:  ex : needed for generating session info.
#                -lo : needed for tracking bad logins
#                 File-related classes : likely all of these could
#                      be relavent for noting sensitive file access.
#                      Could limit to just failures and run the risk
#                      something is not adequately protected or
#                      miss suspicious behavior by an administrator.
#                      This is where some serious tuning and soul
#                      searching would be needed before using in
#                      the real world.

# Description: The script only processes active audit trail files and
#              starts at the beginning of the current audit trail file
#              as designated within /etc/security/audit_data.  This file
#              is monitored with the follow flag of the tail command to
#              process it with a nawk program until the second "file"
#              token is read upon which nawk exits and the shell script
#              attempts to determine the new audit trail file and begin
#              processing it with nawk.  The -d option can be used
#              to have auditing files automatically removed once
#              the audit daemon stops writing to them which can be
#              useful if you do not wish to retain the full fidelity
#              of the raw audit trail files.  As an alternative, you
#              could alter the handle_completed_audit_trail_file()
#              function to invoke a background process to archive
#              or compress old files.  This would not work for any
#              audit trail files created while this script is not
#              up and running.

#              The corresponding audit token lines may be displayed
#              along with the interpreted form by using the -r option.
#              This can be very useful while debugging a problem or
#              trying to understand how BSM works.

#              All audit records can be displayed with the -a option
#              and may be combined with -r to see all audit records
```

```
#               in interpreted and raw form.  Showing all records can
#               help diagnose situations where the expected event
#               is not configured to be audited.

#               Logs can be redirected to syslog with the -s option.

#               Only a select set of audit tokens are currently
#               supported: file, header, path, exec_args, argument,
#               subject, return, attribute, file, text, trailer.
#               A message will be printed if any unsupported tags
#               are processed.

#               The praudit command provided by Sun is used to extract
#               the audit data from the audit trail file.  The binary
#               data could be directly processed which would likely
#               improve performance, but would complicate this code
#               and tie it more tightly to the subleties that may
#               exist within Sun's binary audit data format.

# Prerequisites:
#
#   Sun patches will likely be needed for BSM to work properly.
#   Solaris 2.7 requires patch 106541-16.
#   Bug 4229414 Solaris 7 64 bit BSM auditing with +argv policy break exec()
#
#   To get the most benefit out of this script you need to activate
#   BSM (init 1; /etc/security/bsmconv; init 6) and add to end of
#   /etc/security/audit_startup "auditconfig -setpolicy +trail,argv"
#   The auditconfig command needs to be executed by hand the first
#   time or just reboot afterwards to pick it up.
#
#   Can verify this is in effect with command "auditconfig -getpolicy"

audit_files_base_dir="/var/audit"
export audit_files_base_dir
audit_daemon_info_file="/etc/security/audit_data"
session_output_dir=$audit_files_base_dir
export session_output_dir
hdr_mode="long"

#
# This routine could be customized to archive the old audit trail
# file automatically.
#
handle_completed_audit_trail_file()
{
      # Are we supposed to delete the old audit trail file?

      if [ $delete_old_audit_files -eq 1 -a ! -z "$last_audit_file" ]; then

        # The previous audit trail filename will have been renamed by the
        # audit daemon from something like 20010511132225.not_terminated.generic
        # to 20010511132225.20010511140624.generic, so we alter the filename
        # to substitute a wildcard for the "not_terminated" portion.
        #
        last_audit_file=`echo "$last_audit_file" | /usr/bin/sed -e 's/\.not_terminated\./\.\*\./'`
        /usr/bin/\rm $last_audit_file
      fi
}

get_audit_filename()
{
  last_audit_file="$audit_file"
  iterations=0
  while [ iterations -lt 5 ]
  do

    # Retrieve the name of the currently active audit trail file

    audit_file=`/usr/bin/cut -d: -f2 $audit_daemon_info_file`

    # Should find a new filename since we stopped processing the last one.

    if [ "$audit_file" != "$last_audit_file" ]; then
      handle_completed_audit_trail_file

      if [ ! -f "$audit_file" ]; then
        echo "Error - Could not find the current audit file: $audit_file"
        exit 1
      fi
      return
```

```
    else

      # Check to see if maybe the auditing daemon is down and that
      # is why there is no new audit file identified.

      audit_pid=`/usr/bin/cut -d: -f1 $audit_daemon_info_file`
      nl=`ps -ef | grep $audit_pid | grep -v grep | wc -l`
      if [ nl -eq 0 ]; then
        echo " "
        echo "Auditing daemon is down."
        exit 1
      fi
      echo "No new audit file created yet - waiting..."
    fi
    sleep 3
    iterations=`expr $iterations + 1`
  done
  echo "Error - Could not find the next audit file."
  exit 1
}

# Process command line arguments

usage="$0 [-a] [-d] [-r] [-s]"
delete_old_audit_files=0
show_raw_audit_recs=0
show_all_recs=0
send_to_syslog=0
line_limit=1

while [ $# -ne 0 ]
do
  case $1
  in
    -a)  show_all_recs=1
         ;;
    -d)  delete_old_audit_files=1
         ;;
    -r)  show_raw_audit_recs=1
         ;;
    -s)  send_to_syslog=1
         hdr_mode="short"
         line_limit=80
         ;;
     *)  echo ""
         echo "Invalid option $1"
         echo ""
         echo $usage
         echo "-a   Display all audit trail records."
         echo "-d   Delete audit trail files as audit daemon discontinues"
         echo "     using one."
         echo "-r   Display raw audit token contents"
         echo "     from BSM (may be useful for debugging)."
         echo "-s   Send results to syslog (local1.warning)."
         echo ""
         exit 1
         ;;
  esac
  shift
done

export show_raw_audit_recs
export show_all_recs
export hdr_mode
export send_to_syslog
export line_limit

fieldsep=","
audit_file=""
while [ 1 ]
do

  # Process through the current audit trail file and use the
  # follow (-f) option of tail to wait for new audit records
  # to be created once we catch up.

  get_audit_filename
  /usr/bin/tail -0f $audit_file | /usr/sbin/praudit -d"$fieldsep" | \
     /usr/bin/nawk -F"$fieldsep" \
     'BEGIN { file_beginning=1
```

```
        load_defs()
      }
{

  if (show_raw_audit_recs == 1 && $1 == "trailer")
  {
    raw[++nraw]=$0
  }

  if (($1 == "header" || $1 == "trailer") && "time" in rec)
  {
    # We have reached the end of an audit record, so process
    # it and decide what to do with it.

    analyze_audit_entry()
    delete_audit_rec()
  }

  if (show_raw_audit_recs == 1 && $1 != "trailer")
  {
    raw[++nraw]=$0
  }

  parse_line()

  # Check for the special token indicating the start or end of the
  # audit trail file.

  if ($1 == "file")
  {
    if (!file_beginning)
    {
      print_audit_entry("Auditing to current file has been completed.",std_order_list)

      exit # This is the second file tag in this file which
           # implies the audit daemon has switched to recording
           # in another file.  Exit the nawk script and continue
           # running the shell script to start tailing the next
           # audit file.
    }
    else
    {
      print_audit_entry("Renaming completed audit file to:",std_order_list)
    }
    file_beginning=0
    delete_audit_rec()
  }
}

# Check a single audit event for interesting characteristics
# which might warrant logging

function analyze_audit_entry()
{

  handled=0

  #
  # Scan through a list of critical files we wish to monitor
  # any access to.
  #
  for (i in sensitive_file_list)
  {
    if ("eventid" in rec && rec["eventid"] != "ioctl(2)" &&
       "file" in rec && rec["file"] ~ sensitive_file_list[i])
    {
      print_audit_entry("Access to sensitive file",sens_file_order_list)
      handled=1
    }
  }

  #
  # Extract any commands executed by each user and
  # track their session.
  #
  # Example filenames: session_audtest_2852, session_audtest_2886
  #     where "audtest" is a username and "2852" and "2886" are
  #     session IDs assigned by the system.
  #
  if (("eventid" in rec || "file" in rec) &&
       rec["eventid"] == "execve(2)")
```

```
    {
      sessfile=session_output_dir "/session_" rec["auditid"] "_" rec["sessionid"]
      print rec["time"] ", " rec["file"] ", " rec["cmd"] ", " rec["status"] >>sessfile

      if (show_all_recs || show_raw_audit_recs)
      {
        print_audit_entry("Tracking session",std_order_list)
      }
      handled=1
    }

    #
    # Track bad logins
    #
    # "login - local,login - telnet,login - rlogin,su,rsh,rexec,ftp"
    #
    # Must check returnval instead of status:
    #   rsh access returns returnval=-1, but status="success"
    #     header,119,2,rsh access,,Fri May 11 09:48:53 2001, + 776807943 msec
    #     return,success,-1
    #

    if ("eventid" in rec && rec["eventid"] ~ any_login &&
        "returnval" in rec && rec["returnval"] != "0")
    {
      print_audit_entry("Login failure",std_order_list)
      handled=1
    }

    # Show all other records not already reported if that
    # mode was selected.

    if (show_all_recs && ! handled)
    {
      print_audit_entry("",std_order_list)
    }
  }

  # Print the indicated fields of the current audit event
  # in the order indicated.

  function print_audit_entry(message, order_list          ,i,j)
  {

    if (send_to_syslog == 0) print stars

    # If raw display was selected, show all records
    # in the order they arrived from the audit trail file.

    if (show_raw_audit_recs == 1)
    {
      for (i=1; i <=nraw; i++)
      {
        print "*raw* " raw[i]
      }
      print dashes
    }

    # Make a special copy of the array that we can destroy
    # as the fields have been printed to keep track of the
    # ones remaining.

    for (i in rec)
    {
      print_rec[i]=rec[i]
    }

    # Add the passed in message to the print record structure.

    if (message != "") print_rec["message"]=message

    i=1
    while (i in order_list)
    {
      if (send_to_syslog == 0 &&
          (order_list[i] == "newline" || order_list[i] == "separator"))
      {
        print_line()
        if (order_list[i] == "separator")
        {
          print dashes
```

```
      }
    }

    # Print out all the remaining fields not explicitly requested
    # in no particular order, if this was requested.

    else if (order_list[i] == "remainder")
    {
      for (j in print_rec)
      {
        field=sprintf("%s%s  ", hdg[j,hdr_mode], print_rec[j])
        add_to_line(field)
        delete print_rec[j]
      }
    }
    else if (order_list[i] in print_rec)
    {
      field=sprintf("%s%s  ", hdg[order_list[i],hdr_mode], print_rec[order_list[i]])
      add_to_line(field)
      delete print_rec[order_list[i]]
    }
    i++
  }
  print_line()
  if (send_to_syslog == 0) print dashes
  for (j in print_rec)
  {
    delete print_rec[j]
  }
}

function print_line()
{
  if (line != "")
  {
    if (send_to_syslog == 1)
    {
      system("logger -p local1.warning -t bsmmon \"" line "\"")
    }
    else
    {
      print line
    }
    line=""
  }
}

# Construct line from field to be printed

function add_to_line(field)
{
  if (line_limit != 0 && length(line) >= line_limit)
  {
    print_line()
    if (send_to_syslog == 1) line="     "
  }
  line=line field
}

function delete_audit_rec(         i) {
  for (i in rec)
  {
    delete rec[i]
  }
  for (i in raw)
  {
    delete raw[i]
  }
  nraw=0
}

# Decode information provided in tokens and populate the audit
# record structure.

function parse_line() {
    if ($1 == "header")
    {
      #
      # Example:
      # header,88,2,su,,Wed May 09 09:19:39 2001, + 654011976 mse
```

```
  #
  rec["time"]=$6 $7
  rec["eventid"]=$4
}
else if ($1 == "path")
{
  #
  # Example:
  # path,/etc/security/audit_event
  #
  rec["file"]=$2
}
else if ($1 == "exec_args")
{
  #
  # Example:
  # exec_args,3,
  # ls,-l,junk
  #
  getline
  if (show_raw_audit_recs == 1)
  {
    raw[++nraw]=$0
  }
  rec["cmd"]=$1
  for (i=2; i<=NF; i++) {
    rec["cmd"]=rec["cmd"] " " $i
  }
}
else if ($1 == "argument")
{
  #
  # Example:
  # argument,2,0x5401,cmd
  #
  if ("args" in rec) { rec["args"]=rec["args"] ", " }
  rec["args"]=rec["args"] $2 ":" $3 "(" $4 ")"
}
else if ($1 == "subject")
{
  #
  # Example:
  # subject,audtest,root,root,root,root,404,404,24 7 localhost
  #
  rec["auditid"]=$2
  rec["euserid"]=$3
  rec["egroupid"]=$4
  if ($5 != "-1")
  {
    rec["ruserid"]=$5
  }
  if ($6 != "-1")
  {
    rec["rgroupid"]=$6
  }
  rec["pid"]="PID: " $7
  if ($8 != "-1")
  {
    rec["sessionid"]=$8
  }
  rec["terminal"]=$9
}
else if ($1 == "return")
{
  #
  # Examples:
  # return,success,0
  # return,failure: Not owner,-1
  # return,success,-1  (for "rsh access" - looks like error)
  #
  rec["returnval"]=$3
  colon=index($2,": ")
  if (colon > 1  && colon < length($2))
  {
    rec["statusmsg"]=substr($2,colon+2)
  }

  # Construct our own english status indicator since the
  # one provided is misleading at best.
```

```
      if (rec["returnval"] != "0")
      {
        rec["status"]="failure"
      }
      else
      {
        rec["status"]="success"
      }
    }
    else if ($1 == "attribute")
    {
      #
      # Example:
      # attribute,100555,bin,bin,32,125624,0
      #
      rec["filemode"]=$2
      rec["fileowner"]=$3
      rec["filegrp"]=$4
      rec["fsid"]=$5
      rec["inode"]=$6
      rec["filedev"]=$7
    }
    else if ($1 == "file")
    {
      #
      # Example:
      # file,Tue May 08 15:52:18 2001, + 611461 msec,/var/audit/20010508203916.20010508205218.generic
      #
      rec["time"]=$2 $3
      rec["audfile"]=$4
    }
    else if ($1 == "text")
    {
      #
      # Example:
      # text,successful login
      #
      if ("text" in rec) { rec["text"]=rec["text"] ", " }
      rec["text"]=rec["text"] $2
    }
    else if ($1 == "trailer")
    {
      ; # End of audit record reached
    }
    else
    {
      rec[$1]=$0
      print "unparsed tag: " $0
    }
}


function load_defs()
{
  stars="**************************************************"
  dashes="--------------------------------------------------"

  any_login="^(login -|su$|rsh access$|rexecd$|ftp access$)"

  hdr_mode=ENVIRON["hdr_mode"]
  show_raw_audit_recs=ENVIRON["show_raw_audit_recs"]
  session_output_dir=ENVIRON["session_output_dir"]
  show_all_recs=ENVIRON["show_all_recs"]
  send_to_syslog=ENVIRON["send_to_syslog"]
  line_limit=ENVIRON["line_limit"]

  short="short"
  long="long"
  success="success"
  failure="failure"

  # Labels to use for headings - long and short versions

  hdg["time",long]="Time: ";                  hdg["time",short]="Tm: "
  hdg["eventid",long]="Event ID: ";           hdg["eventid",short]="Evnt: "
  hdg["file",long]="Filename: ";              hdg["file",short]="Fl: "
  hdg["auditid",long]="Audit ID: ";           hdg["auditid",short]="AUID: "
  hdg["euserid",long]="Effective User ID: ";  hdg["euserid",short]="EUID: "
  hdg["egroupid",long]="Effective Group ID: "; hdg["egroupid",short]="EGID: "
  hdg["ruserid",long]="Real User ID: ";       hdg["ruserid",short]="RUID: "
```

```
        hdg["rgroupid",long]="Real Group ID: ";      hdg["rgroupid",short]="RGID: "
        hdg["pid",long]="Process ID: ";              hdg["pid",short]="PID: "
        hdg["sessionid",long]="Session ID: ";        hdg["sessionid",short]="SID: "
        hdg["terminal",long]="Terminal: ";           hdg["terminal",short]="Trm: "
        hdg["status",long]="Status: ";               hdg["status",short]="Sts: "
        hdg["statusmsg",long]="Status Message: ";    hdg["statusmsg",short]="StsMsg: "
        hdg["returnval",long]="Return Value: ";      hdg["returnval",short]="Rtn: "
        hdg["filemode",long]="File Mode: ";          hdg["filemode",short]="Mode: "
        hdg["fileowner",long]="File Owner: ";        hdg["fileowner",short]="Own: "
        hdg["filegrp",long]="File Group: ";          hdg["filegrp",short]="Grp: "
        hdg["fsid",long]="File System ID: ";         hdg["fsid",short]="FSID: "
        hdg["inode",long]="Inode: ";                 hdg["inode",short]="Inode: "
        hdg["filedev",long]="File Device ID: ";      hdg["filedev",short]="DID: "
        hdg["audfile",long]="Audit Filename: ";      hdg["audfile",short]="AudFile: "
        hdg["text",long]="Textual Comment: ";        hdg["text",short]="Txt: "
        hdg["cmd",long]="Command: ";                 hdg["cmd",short]="Cmd: "
        hdg["args",long]="System Call Args: ";       hdg["args",short]="Args: "
        hdg["message",long]="";                    hdg["message",short]=""

        sens_file_order="message,time,separator,eventid,file,auditid,status,statusmsg,remainder"
        split(sens_file_order,sens_file_order_list,",")


std_order="message,time,separator,eventid,auditid,sessionid,status,statusmsg,returnval,cmd,args,file,remainder"
        split(std_order,std_order_list,",")

        # These need to be the actual files and not soft links.
        # If you have hard links, then you need to have the full name
        # of each hard link to be totally covered.  Each filename
        # entry separated by commas is treated as a regular expression.
        #
        sensitive_files="/etc/shadow,/etc/passwd,/etc/group,/etc/vfstab," \
                        "/etc/dfs/dfstab,/etc/inet/inetd\.conf,/etc/.*\.conf"
        split(sensitive_files,sensitive_file_list,",")
      }
    '
done
```

Listing 12 - BSM Monitoring Script (bsmmon)

References

[1] Sun Microsystems, Inc. SunSHIELD Basic Security Module Guide. 2000. URL: http://docs.sun.com/ab2/coll.47.11/SHIELD/ (23 May 2001)

[2] Osser, William and Alex Noordergraaf. Sun BluePrints: Auditing in the Solaris 8 Operating Environment. February 2001. URL: http://www.sun.com/blueprints/0201/audit_config.pdf (23 May 2001)

[3] Sun Microsystems. Sun BluePrints" Scripts and Tools. URL: http://www.sun.com/blueprints/tools/ (23 May 2001)

[4] SRI International - System Design Laboratory. Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD). URL: http://www.sdl.sri.com/projects/emerald/ (23 May 2001)

[5] Networks Associates Technology, Inc. (NAI): PGP Security. CyberCop Monitor for Solaris 2.6 Getting Started Guide Version 2.5. 2000. URL: http://download.nai.com/products/media/pgp/support/cybercop/cyp_mon/UNIXccm25ugs.pdf (23 May 2001)

[6] Internet Security Systems (ISS). RealSecure OS Sensor User Guide Version 5.0. January 2001. URL: http://documents.iss.net/literature/RealSecure/RS_OSSensor_UG_5.5.pdf (23 May 2001)

[7] Holmlund, Daniel and Jeremy Banford. The Linux Basic Security Module Project. 3 December 2000. URL: http://linuxbsm.sourceforge.net/ (23 May 2001)

[8] Department of Defense. Trusted Computer System Evaluation Criteria. 26 December 1985. URL: http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html (23 May 2001)

[9] Endler, David. Intrusion Detection using Solaris' Basic Security Module. 14 July 2000. URL: http://www.securityfocus.com/focus/ids/articles/idsbsm.html (23 May 2001)

[10] Troy, Gene. Common Criteria Project. URL: http://csrc.nist.gov/cc/ (23 May 2001)

[11] Lockier, Paul and David Lodge. Solaris 8.0 Security Release Notes: Common Criteria Certification. 9 January 2001. URL: http://access1.sun.com/contract/EReg/SolarisSolve/solaris8/download/SRN_1.0.pdf (23 May 2001)

[12] Moffat, Darren J. Solaris BSM Auditing. 27 November 2000. URL: http://www.securityfocus.com/focus/sun/articles/bsmaudit1.html (23 May 2001)

[13] Danielsen, Jay. Solaris Auditing. URL: http://www.ameritech.net/users/jayd/Audit.pdf (23 May 2001)

[14] Wenchel, Kevin and Stephen Michaels. Implementing C2 Auditing in the Solaris Environment. URL: http://www.samag.com/articles/2000/0013/0013d/0013d.htm November Supplement SysAdmin magazine. (23 May 2001)

[15] SaberNet.net. Solaris Security Guide. URL: http://www.sabernet.net/papers/Solaris.html (23 May 2001)

[16] Danielsen, Jay. Basic Security Module GUI. URL: http://www.ameritech.net/users/jayd/bsm.html (23 May 2001)

[17] University of California, Davis, Computer Security Laboratory. The Audit Workbench Project. 24 August 1995. URL: http://seclab.cs.ucdavis.edu/awb/AuditWorkBench.html (23 May 2001)

[18] Boran, Seán. Solaris C2/BSM security notes - Draft. 30 March, 2001. URL: http://www.boran.com/security/sp/Solaris_bsm.html (23 May 2001)

[19] Bezroukov, Nikolai. Softpanorama University Unix Security Pages: Solaris Hardening and Security. URL: http://www.softpanorama.org/Security/sos.shtml (23 May 2001)

Assignment 3 - "Analyze This" Scenario

The snort logs provided begins on January 20, 2001 at 12:22 am and ends on March 12, 2001 at 11:58 pm. An initial analysis of the data was performed by extracting the time interval covered by each file which revealed that three files were duplicates. SnortA36.txt, UMBCNI5.txt, and UMBCNI31.txt were all eliminated as exact copies of, respectively, SnortA35.txt, UMBCNI3.txt, and UMBCNI25.txt.

Three types of logs were available for analysis:

- Alert
- Scan
- Out of Specification (OOS)

The spp_portscan messages for the scan status and end of scan were excluded when extracting lists of the machines and ports appearing the most within the traffic provided, because the port scan messages from the beginning of the scan can be used to represent each scan alert in gathering statistics.

Most Prevalent Source Addresses

IP Address	Name	Number of Packets
155.101.21.38	bonfire.crsim.utah.edu	81304
MY.NET.218.90		34545
171.69.248.71	tower-u1.cisco.com	29058
140.142.19.72	netfx.uwtv.washington.edu	28117
MY.NET.150.220		24219

bonfire.crsim.utah.edu (155.101.21.38)

This machine is managed by the University of Utah and was the largest source of packets by more than double the nearest challenger. Investigation revealed that this address is within the Combustion and Reaction Simulations group within the Department of Chemical and Fuels Engineering. A cached web page from April 6th within the Google search engine for the URL http://msd.cdt.luth.se:8000/all/ entitled "Current mStar Sessions" mentions this particular IP and hostname. The page indicates that as of April 6th, this machine had been heard from. It indicates that this server was providing MPV encoded video and MPA encoded audio. This source is subsequently no longer listed on the current version of this web page.

Examining the data sent by this source reveals that it exlusively is sent as UDP packets to 224.2.127.254 port 9875. This address is a multicast address which means that normally this type of traffic can be thought of as being broadcast to anyone who cares to receive it. The Multicast Backbone (MBONE) FAQ at http://www.cs.columbia.edu/~hgs/internet/mbone-faq.html indicates that Global announcements of audio and video transmissions over MBONE are sent this this destination IP and port.

The source port used in all of these packets is very stable, but not completely constant. From January 30 until February 28th at 3:21 pm, a source port of 1037 was exclusively used and 57280 packets fall into this time period. Only three minutes later, a packet arrives with a new source port of 1027 and then is consistently used in the next 24024 packets until March 10th. This seems to suggest that the program used to send this traffic may have crashed or restarted at this transition time.

This traffic arrived at all hours of the day and night, but did increase between the hours of 3 am - 11 am. It appears that this traffic was captured within Snort because it was exchanged between IP addresses that were each outside our internal network according to its configuration. No traffic was logged that originated within our network that was addressed to this particular machine.

This all seems to support the view that this is merely multicast traffic that appears to be harmless.

MY.NET.218.90

This machine seems to be originating a large amount of UDP traffic with a source port of 1036 destined for many different external addresses.The traffic seems to be distributed across a number of addresses such that the most frequent destination, 216.19.133.116 (within a network block EC08-1 manged by Exodus Communications Inc.), only received 2041 packets out of the total 34545 sent. This destination was only contacted within a two hour period on Sunday, February 4th beginning at 5:06 pm. No clues on the Internet as to what this destination machine might be used for typically. The destination ports used varied considerably which seems to imply these may be going to ephemeral ports or there is port scanning being performed. All of the traffic logged as sent from this machine during an 8 hour period that Sunday with 2835 unique destinations contacted with a total of 34545 packets. Traffic was most intense between 11 am - 2 pm and 5 pm - 8 pm.

The only suspicious packet arriving at MY.NET.218.90 prior to this potential scanning was a SYN FIN packet received on January 21st at 2:45 pm from sumac.asrc.cestm.albany.edu (169.226.202.234).

Port 1036 could be some sort of rpc managed program like mountd or just an ephemeral port on MY.NET.218.90. It is not clear whether this machine is the server or client in this particular traffic. Due to the odd time of this traffic (Sunday) and its intensity, further investigation is recommended to check for compromise or inappropriate use of this machine.

tower-u1.cisco.com (171.69.248.71)

This machine sent 29058 packets of multicast traffic that was logged exclusively arriving at 224.2.127.254 UDP port 9875. This seems to be video or audio traffic and is shown as "Places All Over the World at 128K" by the web pages "SDR Global Session Monitoring Effort" at http://imj.ucsb.edu/sdr-monitor/ and "SDR Session Chart" at http://www.aciri.org/fenner/mcast/sdr-chart.html. One of the web pages corroborates the multicast IP and port seen in our traffic.

No messages or reponses were logged from our internal network back to this source, which is what we would expect. The source port was once again fairly constant similar to the other multicast traffic analyzed. From January 30th until February 4th, the source port was 41720. When the next packet was logged on February 6th, the source port changed to 45161 and remained as such through the remaining packets on March 10th.

The packets were received at all hours of the day and night, but did increase between 2 am and noon when viewed overall. Slight increases were also seen during the hours of 6 pm, 8 pm, and 10 pm.

This traffic is thus very likely to be more multicast traffic that is benign.

netfx.uwtv.washington.edu (140.142.19.72)

This machine sent 28117 UDP packets of multicast packets to 224.2.127.254 port 9880. No messages were logged as destined back to this address from our internal network as we would expect. This is likely more benign multicast network traffic.

MY.NET.150.220

Of the total 24219 packets logged as originating from this machine, 21961 packets were directed toward UDP port 28800. Investigation of the uses of this port indicate it is commonly during participation in online gaming (i.e. MSN Gaming Zone, Starsiege Tribes, MechWarrior: Vengeance). This activity occurred between February 6th and March 6th and was limited to between the hours of 1pm and midnight exclusively. The peak usage was within 8pm to midnight. No traffic was logged as being sent to this machine. The source ports for all of this traffic except for 4 packets was also UDP port 28800 with a destination of 212.130.43.38 (Netname DK-EUROCONNECT-990113). The remaining 4 packets all came from source port 4793 and were destined for 3 distinct IP addresses within one minute beginning at February 7th at 9:35 pm: 63.161.101.226 (netname NETBLK-SPRINT-3FA164-1), 65.26.197.237 (managed by Road Runner-Central), and 148.235.117.164 (NETBLK-UNINET-NET10). No other traffic was exchanged with these three addresses in the logs. The majority (21961) of the packets from this source had a destination port that was also 28800. The next most used port was 1086 with 89 packets. This usage should be investigated further to ensure it is benign and consideration of updating the accepted system usage policy if necessary.

Most Prevalent Destination Addresses

IP Address	Name	Number of Packets
224.2.127.254	sap.mcast.net (multicast address)	360172
129.2.246.94	Within student.umd.edu domain - University of Maryland	21060
233.28.65.197	(multicast address)	20713
233.28.65.255	(multicast address)	17397
224.0.1.41	gatekeeper.mcast.net (multicast address)	12036
MY.NET.160.109		10000
MY.NET.60.8		8848

sap.mcast.net (224.2.127.254)

161 different IP addresses were seen sending UDP traffic to this multicast address. The 224.2.0.0/16 address range is used for audio and video conferencing sessions according to http://cvw.sourceforge.net/cvw/faq/avmcast.php3. All of this traffic only goes to ports 9875, 9880, and just two packets to destination port 0. These two strange packets were sent from 140.116.247.125 (nba23.csie.ncku.edu.tw) with a source port of 9875 and a destination port of 0 during the same second on February 28th at 6:41 am. This machine, nba23, did exchange 139 other packets within our logs. The two destination port 0 packets are the first noted from this source since February 23rd. More normal packets arrive from this source to the normal destination port of 9875 two and a half minutes after the ones with a destination port of 0. Perhaps this behavior is normal for MBONE. One set of messages on the Internet (http://lists.bell-labs.com/pipemail/sip/2000q3/001486.html) suggests that a port of 0 and the same SDP should be sent to delete an existing media stream. It is not obvious at this time whether this directly relates to these particular packets. These were the only packets addressed to the multicast addresses in the logs with a dest port of 0.

There is an older vulnerability (VN-99-03) on CERT that is in the SDR package which could affect anyone using version 2.6.2 or older (http://www.cert.org/vul_notes/VN-99-03.html). Any site listening for session advertisements could be affected. If SDR is being utilized within this network, it would be wise to ensure a more recent version is being utilized in case these packets are of a more insidious nature.

Unnamed address within student.umd.edu domain (129.2.246.94)

The next runner up was 129.2.246.94, an unregistered address within the student.umd.edu domain for the Universidy of Maryland, with 21060 packets.

All of these packets correspond to an intense set of tcp port scans initiated with SYN packets from MY.NET.221.26. Looking for other traffic from this same source within our logs only turns up one additional packet directed toward 24.48.226.183 (part of a block of addresses managed by Adelphia Cable Communications) and a destination port of 1799 from the signature Ramen and SubSeven source port of 27374 at 11:18 pm on Sunday, February 11th. A machine infected with Ramen would establish an http server at port 27374, so this could imply that MY.NET.221.26 has the Ramen worm. The port scans occurred the day before, Saturday, February 10th, from 3:19 am to 2:04 pm.

The first scan lasted just five minutes starting at 3:19am with 1146 packets. The ports tried ranged from 1 - 2743 with not all ports attempted. The second scan began at 3:54 am and lasting 28 minutes with the most packets sent of the three and more than double of the nearest other scan with 14184 packets. The destination ports tried were in the interval 1 - 28706, but did not try all of the ports in this range. The last scan started at 2:00 pm and lasted five minutes again with 5730 packets. This scan does exhaustively hit all ports and hits some multiple times. In each of the separate scans, no single port was hit more than three times which likely corresponds to retries. In each of the scans not all of the ports within the range were tried, but they all started at port 1.

The second scan spends about 2-3 seconds scanning ports in ascending order skipping anywhere from none to 14 ports as it increases, before starting cycling back to earlier ports which were not tried the last cycle and working forward again in ascending order once more. This pattern seemed to continue to repeat itself.

The last scan was more intense in the rate of packets issued with an rough average of 1300 packets per minute versus only 300 packets per minute during the first couple of scans.

MY.NET.221.26 should be examined for possible infection with the Ramen worm.

Multicast address (233.28.65.197)

The multicast address 233.28.65.197 had 20713 udp packets sent to it from 206.190.54.67 (Yahoo! Broadcast Services) and source port 1030 with a destination port of 5779. This traffic is likely benign.

MY.NET.160.109

Looking at the 10000 packets that went to MY.NET.160.109 reveals that all but 8 were udp from 206.112.192.106 (One Net Communications network block). shell.one.net (http://www.the-project.org/routig/0898/msg00017.html) This machine is their server to support their users with a shell account (http://www.one.net/network/numbers/). The destination port numbers were across a wide range and makes this look like a UDP port scan. No traffic was logged as sent to 206.112.192.106. The logs show it happened on Tuesday, January 30 starting at 8:40 pm and 44 seconds and lasting until 8:41 pm and 22 seconds. So this scan lasted 38 seconds. This information was derived from the scan log SnortS7.txt. The source ports did not vary much during the scan with only three unique values. The majority and first 8571 of the packets used a source port of 1676. The next 7 packets used source port 1678. The last 1414 packets used source port 1679. Analyzing the times and destination ports with this interesting set of source ports does not shed any light on what might be the reason. All of the transitions between the source ports happens during the same second.

The destination ports are spread almost over the entire possible set of port numbers ranging from 1 through 64987 with gaps throughout. An analysis of the distribution shows it is fairly uniformly distributed and does not seem to show any particular intelligence as to which ports to try.

The snort alert log (SnortA6.txt) corroborates the information retrieved from the scan log other than the times,which I believe correspond to when the scan detection code generated the alerts. It shows a udp portscan from 206.112.192.106 beginning at 8:53 pm and 21 seconds and ends at 8:54 pm and 6 seconds to one host with 9992 packets. So this lasts 45 seconds. So this scan shows it starts almost 13 minutes later than the one from the scan log. The other possibility is that the spp_portscan code within Snort introduced the inaccuracies. Examining the source code within spp_portscan.c indicated some difficulties in tracking statistics, but did not mention specifically any issues with time and seemed to imply more trouble in tracking packets stats. So another possibility is that the number of packets reported within the portscan messages within the alert log are incorrect. This also seems unlikely because the individual intermediate portscan status messages contain a count of packets since the last status message and the total of these equals the final reported number of packets (9992) shown in the end of portscan message.

Laurie@edu reported three errant UDP packets on November 30, 2000 against a single host from 206.112.192.106 (http://www.sans.org/y2k/120500.htm).

It is recommended that the network defenses be examined to verify that only services needed are permitted. Additionally, MY.NET.160.109 should be examined to ensure it successfully weathered this possible attack. The configuration of this machine should be scrutinized to see if any unecessary network services can be eliminated and if any further host-based protection can be employed. It may be prudent to relocate this machine to a new IP address since this seemed to be a targeted attack.

MY.NET.60.8

A total of 8848 packets were destined for MY.NET.60.8. Thirty six different machines were responsible for sending these packets, but the majority (8642) came as tcp SYN packets from 24.141.226.62 (d141-226-62.home.cgocable.net) on Saturday, February 10 from 12:05 pm until 12:22 pm. The ports probed were within the range 1 - 9998. No other packets were received from this originator within the logs and no packets were returned as far as the logs show. This machine is within a network block managed by Cogeco Cable Solutions based in Burlington, Ontario, Canada which appears to provide Internet service over cable modems among other things.

96 packets were received as well from 24.64.75.243 (within the cg.shawcable.net domain) within a network block managed by Shaw Fiberlink Ltd. of Calgary, Canada for cable modems on Thursday, March 1 at 9:02 pm and lasting for just 19 seconds. No other packets were received within our network from this source or returned to it according to our logs. The ports hit within the range from 5-32772 and each of the ports were attempted only once.

A set of 42 TCP SYN packets were received at this same destination MY.NET.60.8 from 165.247.5.203 (user-2ive1eb.dialup.mindspring.com) within a network managed by EarthLink on Saturday, February 24 at 2:37 pm for 20 seconds. The alert log also corroborates these 42 packets but once again there is a delay of fifteen minutes from the times indicated in the scan log similar to another case just noted previously. This scan appears within a different set of files than the other case (UMBCNI35.txt). Due to the unlikelihood of two different scans having these same characteristics and not representing the same scan, I have come to the conclusion that these 42 packets indicated by the scan and alert logs correspond to one another despite the differences in times. No destination ports were repeated and they ranged from 60 - 3985 with the exception of port 32780. Port 2005 is shown as attributable to the TransScout trojan. No other packets were received from 165.247.5.203 and no packets sent back according to the logs

MY.NET.60.8 should be examined to make sure it withstood this attack, its defenses scrutinized to be sure they are adequate and current, and possibly relocate this machine to a new IP address due to the targeted nature of this attack.

Most Prevalent Source Ports

Source Port Number	Number of Packets
27888	139432
28800	132429
1037	60968
13139	55614
1036	49280

27888

UDP port 27888 appears to be a port relating to online gaming (Shogo, No One Lives Forever). The destination ports seem to range from 1025 all the way up to 65456. This implies that these are likely ephemeral ports and these messages are responses. Only 5 messages were sent inward into our network with those all going to MY.NET.178.154. The remaining 139427 packets were leaving our network toward 1516 different external hosts. These messages are being sent from 34 distinct machines all of which are internal with the exception of one. This traffic went at all hours except between 5 am and 11 am. The peak usage was 2 pm - 6 pm, 7 - 8 pm, 10 - 11 pm, and 1 am - 2 am. The hours of this traffic seems most disturbing and seems to imply it may be related to a daemon process serving external requests.

The MY.NET network may very well be hosting game servers for others to contact. Regardless, this activity seems not likely be of a business-related nature. It should be considered to configure or install equipment such as routers and firewalls to restrict traffic such as this unless it turns out to be authorized. Attempting to severely limit network traffic to a very few number of critical services is highly recommended to ensure that MY.NET is protected against future threats.

28800

UDP port 28800 may indicate participation in online gaming (Starsiege Tribes, MechWarrior: Vengeance, MSN Gaming Zone). This traffic went at all hours except between 6 am and 11 am. The peak usage was 1 pm - 3 pm, 5 - 7 pm, and 9 - 11 pm. The same comments apply from the previous port. This traffic may correspond to game servers setup for others to access.

1037

The majority of this traffic appears to be destined for multicast addresses. There are 57281 packets logged as sent to 224.2.127.254 and comes from 155.101.21.38 except for one packet from another external address. Additionally 3585 packets logged as sent to 233.28.65.255 from 63.250.208.139. The remaining traffic is so low that it very likely corresponds to standard ephemeral port usage. This traffic is distributed over all 24 hours although it does peak between 3 am and noon. This traffic is likely benign.

13139

There were 55614 UDP packets shown to come from source port 13139. Of these, the majority (51948 packets) are sent to destination port 13139. This traffic between the same port numbers was at all hours, but quiets down considerably between 2 am and 11 am. The next most prevalent destination port number was 1025 for this traffic and it seems to travel from MY.NET to external addresses with no particular machine appearing in a significant way. Both of these cases had similar trends to one another which seems to boost the supposition that they are related to one another. When examining all of the traffic using a source port of 13139, it seems to emanate from MY.NET. There are many machines involved with MY.NET.220.142, MY.NET.214.250, and MY.NET. 223.246 as the leaders with many more close behind.

This port is used with some online gaming. The Game Spy Arcade uses this port for custom UDP pings (http://www.gamespyarcade.com/software/support/firewalls.shtml). This may be additional gaming-related network traffic and the same comments apply as the other gaming ports.

1036

The large majority of this traffic is also multicast-related and appears to likely be benign.

Most Prevalent Destination Ports

Destination Port Number	Number of Packets
9875	327801
28800	135330
7778	62138
21	53158
13139	51951

9875

In checking the destinationof these messages, they are almost all directed toward the multicast address of 224.2.127.254. This traffic is likely benign.

28800

This UDP traffic was established as likely to be gaming related when investigating the source ports previously. The packets appear to go in one particular direction. They seem to flow from MY.NET to external addresses. There is no predominant recipient, but there do appear to be a few hosts within MY.NET that send much more than other MY.NET hosts. Although many MY.NET hosts are sending such traffic. The MY.NET.150.0 class C network seems to be represented significantly within the hosts sending such packets. MY.NET.150.220, MY.NET.150.133, MY.NET.150.225, and MY.NET.150.143 are the top four machines generating this over the network. MY.NET.150.145 and MY.NET.150.41 are also prominently featured.

Recommend verifying that such activity is reasonable and embarking on a large effort to rectify configurations should it not be.

7778

This UDP traffic is commonly associated as a query port for the game Unreal Tournament. These packets also seem to flow from within MY.NET toward external addresses. The top senders are MY.NET.204.150, MY.NET.217.142, and MY.NET.209.238. When examining the logs for IP addresses listed on an Unreal Tournament web page (http://www.virtualand.net/qnews/serv5.html), we find a large number of these present. Some of the destinations matched are:

Destination IP Address	Number of Packets

200.212.26.211 (ut.hardmob.com.br)	172
200.249.220.21	88
200.185.12.20 (ut.lagzero.com)	28
200.185.12.19	36
200.185.12.17 (demo01.lagzero.com)	23
212.76.32.50	357

Same gaming policy recommendations apply here as well.

21

Of the packets addressed to a destination port of 21, all but 83 are TCP and a large number (48631) come from a source port also of 21. Of great concern when examining these packets are the 18782 packets coming from 130.234.184.112 (termos.keltti.jyu.fi), which managed by the University of Jyvaskyla Computing Center in Jyvaskyla, Finland. These packets ran across MY.NET extensively and very quickly. This scan happened on Sunday, February 25th from 4:50 am until 5:24 am. Most of the packets from this attacker were of the SYN FIN variety which shows some effort to evade protection devices. No correlations were found for this machine conducting other attacks of any kind.

The next runner-up, 169.226.202.234 (sumac.asrc.cestm.albany.edu), managed by the University at Albany - State University of New York with 12129 entries within the logs also performed a wide sweeping scan of ftp ports within MY.NET with SYN FIN packets. This scan occurred on Sunday, January 21st from 2:26 pm until 2:48 pm. There are a couple of packets listed against this originator on dshield.org aimed toward TCP ports 20 and 24 on April 19th (http://www1.dshield.org/ipinfo.php?ip=169.226.202.234).

There were no predominant recipients of ftp traffic versus others within the logs. This is likely due to the fact that most of the activity logged related to ftp port scans.

Incoming ftp traffic should be restricted with routers and firewall to only allow it to reach authorized public ftp servers. Machines that must be public ftp servers should be hardened with the latest ftp daemon patches and limit other services these machines provide, if possible.

13139

The same comments apply here as they did with this port as a common source.

Most Prevalent Alerts

Type of Alert	Alert Message	Number of Packets
UDP Outside	[**] UDP SRC and DST outside network [**]	436882
Watch List	Watchlist 000220 IL-ISDNNET-990517	15021
Port Scan	spp_portscan PORTSCAN DETECTED	11656

UDP Outside

There were 31 distinct multicast addresses that packets were addressed to totaling 423321 packets. The majority of the multicast packets (360172) by far were addressed to 224.2.127.254.

After excluding the multicast destinations, only 13561 packets remain with the top three destinations being all broadcast addresses. The top broadcast address destination involves 3867 packets originating from 10.0.0.1 toward 10.255.255.255. The thirdmost non-multicast destination was 192.168.0.255, addressed by 1306 packets, which were all coming from 192.168.0.2 except for one packet from 192.168.0.1. All of these addresses are private RFC 1918 IP addresses which are not routable to the Internet. This broadcast address covers this entire private address space. This particular traffic occurred many days beginning on February 20th at 12:12 am and running until March 10th at 11:01 pm. This traffic occurred at all times of the day and night. All of the broadcast traffic for the 10 network is addressed to port 67 (bootps). This implies these messages may be destined for a BOOTP server with an all-subnets broadcast, but usually these type of messages are sent to the local limited broadcast address of 255.255.255.255. This implies that the traffic is likely being distributed across routers. The 192.168.0 network is exclusively addressed to and from only ports 137 and 138 (both netbios). The 192 traffic is consistent with Netbios-related packets.

The secondmost non-multicast destination was 169.254.255.255 and these were seen between January 30th through March 10th. This traffic happened at all hours like the others discussed previously. This traffic was also netbios related being exchanged to and from ports 137 and 138. All 331of the machines sending this traffic were within the 169.254 class B network. In researching this set of addresses, it was discovered that these are assigned by IANA for Automatic Private IP Addressing (APIPA) which can be used when a DHCP server is not available so that a machine can assign itself an IP address. These addresses are similar to the others just discussed in that they are not routable to the Internet.

The legitimacy of this traffic should be investigated with your network engineers, just to be safe. Regardless, it is recommended that some sort of firewalling device or router for the entire network be configured with anti-spoofing filters. That is, only traffic originating with the MY.NET addresses can leave the network and only packets addressed to MY.NET addresses can enter the network. Additional exceptions may be required for other traffic, such as multicast, should it be necessary. Filters such as this may already be in place, but it is worth being sure. It is not clear whether the sensor detecting this traffic is inside or outside any security perimeter which may exist.

The fourthmost prevalent destination is 162.129.112.40 (jhwins01.jhoc1.jhmi.edu) which is within the netblock managed by Johns Hopkins Medical Institutions. All of this traffic, 656 packets, came from 162.129.159.138, which is also within the same netblock and does not resolve to a name, involved only port 137 for the source and destination.

Since it seems suprising to see these packets which would be expected to be limited to networks within Johns Hopkins and not within our networks, we will further examine other traffic involving the class B netblock 162.129. There were an additional 11 packets destined for this netblock and all of them except one originated from 162.129.159.138. All of them are UDP packets. The one other packet came from MY.NET.213.246 source port 2788 destined for 162.129.49.154 port 6346 with the SYN flag set and happened on February 27th at 5:14 pm. Port 6346 is commonly Gnutella which is a file sharing program which may not be appropriate depending upon your network usage policy. The other packets from 162.129.159.138 were sent to 162.129.112.31 (4), 162.129.16.39 (3), 162.129.112.30 (3). The three of these packets destined for 162.129.16.39 only involved port 137 for the source and destination. The remaining seven packets are a little more interesting and come in on port 137, but are

addressed to port 53 (DNS). I suspect that there is nothing more nefarious here than a Windows-based machine doing name resolution. Research indicates that Windows-based machines use a source port of 137 when doing DNS lookups. To complete the search, it was verified that only packets from 162.129.159.138 were among those addressed from within this netblock that was captured within our logs.

The presence of the John Hopkins traffic is not obviously of ill intent, but it does seem something curious that should be investigated further. Perhaps there is a special affiliation between your organization and Johns Hopkins. The traffic was only present within the logs from February 3rd until February 27th.

ISDN Net Watch List

This traffic is of note due to its origin from the ISDN Net based in Israel. The logs contain alerts relating to this from January 30th until March 10th. There were 53 different IP addresses from ISDN Net that were noted and combined these addressed 78 machines within the MY.NET network. The largest number of packets (4061) came from 212.179.41.169 (fr-c41169.bezeqint.net) and they all went to MY.NET.213.250 and all with a destination port of 6688 and a source port of 1113 between 7:01 pm and 9:35 pm on Saturday, March 10th. The port 6688 is commonly used as the Napster (file sharing program) client port.

All 2186 packets from 212.179.21.179 were directed to MY.NET.207.226 with a destination port of 6699 (commonly Napster) and a source port of 1172 on Tuesday, February 6th between 6:11 am and 8:04 am.

There were 62 packets logged as addressed toward this watch list from our network. The most predominant originator was MY.NET.150.143 with 24 packets and MY.NET.98.176 with 7 packets. All of these and the majority of the others went to ports 28800 and 13139 which are commonly gaming ports.

This traffic may in fact be predominantly gaming and music file sharing. These activities are not to be encouraged from a security perspective and it would be advisable to prohibit such activities due to the possibility of these actions causing a security incident. It would be recommended to update the configuration of the firewalls and routers to be more restrictive and deny everything except what is explicitly allowed.

Port Scans

MY.NET.70.38 was the most prolific in the number of port scans noted in the logs with 3538. The nearest machine, MY.NET.140.21, had only 245 entries.

Almost every single log entry for packets originating from MY.NET.70.38 come from just four ports: 36327, 36338, 36339, and 36340. The destination ports vary significantly and are distributed with no particular machine being the destination for more than a few packets. DNS was by far the most often used destination port with 1269 TCP packets. The closest runnerup was port 36329 with 14 packets. The packets not directed to TCP DNS were distributed evenly between 30000 and 45000. The activity originated from this host occurred between February 20th and February 23rd. No packets are in the logs which are addressed back to MY.NET.70.38 during this four day period. The packets emanating from this machine are very odd with all sorts of interestingly invalid flags set. Many different IP addresses used for the destinations such that there really isn't a significant one that rises to the top.

When considering this traffic hour by hour across these four days, the peak times were February 20th from 5 am - 8 am and February 23 from 7 am - 8 am. There were 6002 SYN packets with a source port predominantly set to 36338, 5470 Xmas (FIN, PUSH, and URGENT flags set) packets always with a source port of 36340, 4786 NMAP TCP ping packets with a source port of 36339, and 4134 UDP packets with a source port of 36327 noted in the logs from this machine.

MY.NET.70.38 was logged with 4786 NMAP TCP pings. TCP pings are sending an unsolicited ACK packet to a machine and waiting for an expected RESET if the machine is up. This is an alternative to normal ICMP echo requests which are sometimes filtered by firewalls. This rule checks that the acknowledge number is a 0 which is a special characteristic of NMAP in this case. In all of these packets the source port was always 36339, but the destination ports varied.

Looking at a subset of the packets in time order, one thing that stands out is that the destination machines are contacted in ascending order based upon their IP address. Not each address is attempted, but it only skips a small number of IPs and is usually consecutive. The types of packets varies, but there are usually a few of them directed toward each probed host. Most of the SYN packets are directed toward the DNS service.

This does appear to likely be an intense scanning operation directed against our internal network from one of our internal machines. The types of probes seems to indicate possibly an attempt to determine which machines are available (TCP pings), the type of Operating System being run, and whether it is a DNS server or perhaps even a zone transfer attempt since the DNS contact was done over TCP. The scan started with MY.NET.96.32 and went through MY.NET.255.249 and covered 66 class C networks within the MY.NET network.

The machine MY.NET.70.38 should be scrutinized to see if there may be some sort of compromise or inappropriate usage.