Global Information Assurance Certification Paper

GIAC Intrusion Detection Curriculum – Practical Assignment

SANS Parliament Hill 2000 – 21-14 August 2000

Marc Grégoire

Assignment 1 – Network Detects

Detect 1

19:52:16.700562 P 209.221.143.119 > MY.NET.0.0: icmp: echo request (ttl 236, id 40308)
19:52:16.702422 P 209.221.143.119 > MY.NET.1.0: icmp: echo request (ttl 236, id 40310)
19:52:16.704101 P 209.221.143.119 > MY.NET.0.255: icmp: echo request (ttl 235, id 40309)
19:52:16.704846 P 209.221.143.119 > MY.NET.1.255: icmp: echo request (ttl 235, id 40311)
19:52:16.722646 P 209.221.143.119 > MY.NET.10.0: icmp: echo request (ttl 236, id 40312)
19:52:16.723225 P 209.221.143.119 > MY.NET.100.0: icmp: echo request (ttl 236, id 40314)
19:52:16.724688 P 209.221.143.119 > MY.NET.10.255: icmp: echo request (ttl 235, id 40313)
19:52:16.743052 P 209.221.143.119 > MY.NET.101.0: icmp: echo request (ttl 236, id 40316)
19:52:16.744661 P 209.221.143.119 > MY.NET.101.255: icmp: echo request (ttl 235, id 40317)
19:52:16.745180 P 209.221.143.119 > MY.NET.100.255: icmp: echo request (ttl 235, id 40315)
19:52:16.765283 P 209.221.143.119 > MY.NET.102.255: icmp: echo request (ttl 235, id 40319)
19:52:16.779988 P 209.221.143.119 > MY.NET.102.0: icmp: echo request (ttl 236, id 40318)
19:52:16.780242 P 209.221.143.119 > MY.NET.103.0: icmp: echo request (ttl 236, id 40320)
19:52:16.782285 P 209.221.143.119 > MY.NET.104.0: icmp: echo request (ttl 236, id 40323)
19:52:16.784679 P 209.221.143.119 > MY.NET.103.255: icmp: echo request (ttl 235, id 40322)
19:52:16.785161 P 209.221.143.119 > MY.NET.104.255: icmp: echo request (ttl 235, id 40324)
19:52:16.803304 P 209.221.143.119 > MY.NET.106.0: icmp: echo request (ttl 236, id 40327)
19:52:16.803445 P 209.221.143.119 > MY.NET.105.0: icmp: echo request (ttl 236, id 40325)
19:52:16.805362 P 209.221.143.119 > MY.NET.105.255: icmp: echo request (ttl 235, id 40326)
19:52:16.822769 P 209.221.143.119 > MY.NET.107.0: icmp: echo request (ttl 236, id 40329)
19:52:16.825079 P 209.221.143.119 > MY.NET.106.255: icmp: echo request (ttl 235, id 40328)
19:52:16.825779 P 209.221.143.119 > MY.NET.107.255: icmp: echo request (ttl 235, id 40330)
19:52:16.842345 P 209.221.143.119 > MY.NET.109.0: icmp: echo request (ttl 236, id 40333)
19:52:16.842578 P 209.221.143.119 > MY.NET.108.0: icmp: echo request (ttl 236, id 40331)
19:52:16.845152 P 209.221.143.119 > MY.NET.108.255: icmp: echo request (ttl 235, id 40332)
19:52:16.862349 P 209.221.143.119 > MY.NET.11.0: icmp: echo request (ttl 236, id 40335)
19:52:16.864681 P 209.221.143.119 > MY.NET.109.255: icmp: echo request (ttl 235, id 40334)
19:52:16.865246 P 209.221.143.119 > MY.NET.11.255: icmp: echo request (ttl 235, id 40336)
19:52:16.882038 P 209.221.143.119 > MY.NET.110.0: icmp: echo request (ttl 236, id 40337)
19:52:16.883332 P 209.221.143.119 > MY.NET.111.0: icmp: echo request (ttl 236, id 40339)
19:52:16.884518 P 209.221.143.119 > MY.NET.110.255: icmp: echo request (ttl 235, id 40338)
19:52:16.903157 P 209.221.143.119 > MY.NET.112.255: icmp: echo request (ttl 236, id 40342)
19:52:16.904342 P 209.221.143.119 > MY.NET.111.255: icmp: echo request (ttl 235, id 40340)
19:52:16.904604 P 209.221.143.119 > MY.NET.112.0: icmp: echo request (ttl 235, id 40341)
19:52:16.923835 P 209.221.143.119 > MY.NET.113.0: icmp: echo request (ttl 235, id 40343)
19:52:16.924658 P 209.221.143.119 > MY.NET.114.0: icmp: echo request (ttl 235, id 40345)

1. Source of trace (Detect 1)

This traffic is from my network. It is sniffed at the main gateway to the Internet.

2. Detect was generated by: (Detect 1)

Shadow intrusion detection system. This Shadow system (version 1.6) is setup as an experimental facility. It was setup to collect the raw network traffic, to be later used by analysts experimenting with filters. This trace was generated by me using tcpdump filters on the raw traffic data archived by the Shadow system. In this particular case, I was looking for malicious ICMP activities. I filtered on ICMP traffic, then on "echo request" messages, and finally on a specific host that seemed involved in scanning my network. The trace is in a tcpdump output format. Here is an ICMP sample with description:

Sample:
19:52:16.704101 P 209.221.143.119 > MY.NET.0.255: icmp: echo request (ttl 235, id 40309*)*

Description:
19:52:16.704101 = timestamp (hh:mm:ss.ssssss)
P = this characters always shows up in my output, it is apparently an artifact of the Linux flavor used. It is not a protocol flag (not a PUSH). Not used for the analysis.
209.221.143.119 = source IP address
MY.NET.0.255 = destination IP address (sanitized)
icmp: echo request = the ICMP message
ttl 235 = Time To Live value
id 40309 = Packet Identification number

3. Probability the source address was spoofed (Detect 1)

The source address is not likely to be spoofed. This trace has a Smurf (ICMP broadcast) flavor, which would normally point to a spoofed source, but it also has a reconnaissance flavor, i.e. scanning hosts. Normally, in reconnaissance, the source address is not spoofed because the source host needs to see if the scanned hosts reply and therefore reveal information. In this case, I investigated the source address and found out that it is an organization scanning the net, building a database of Smurf-amplifiers (http:\\www.netscan.org). Their goal is apparently to detect the "misconfigured" hosts so that they can be fixed and not be used as Smurf amplifiers. If the netscan.org address was spoofed, let say by hackers who disagree with netscan.org "noble" goal of reducing Smurf attacks, we would have probably seen destination addresses that are part of netscan.org database, i.e. confirmed Smurf amplifiers, which is not the case for my network.

4. Description of attack: (Detect 1)

ICMP Echo Request sent to x.y.z.0 and x.y.z.255 broadcast addresses on my network.

Reference: Common Vulnerabilities and Exposures

CVE-1999-0513	ICMP messages to broadcast addresses are allowed, allowing for a Smurf attack that can cause a denial of service.

My network is also scanned with these ICMP stimuli for reconnaissance purposes. So the attack can also be viewed as a probe for Smurf-amplifiers.

5. Attack mechanism: (Detect 1)

Sending a single ICMP echo request message to x.y.z.255 or to x.y.z.0 (broadcast addresses) could generate replies from all the individual hosts on the x.y.z subnet. It has an amplification effect. This technique (Smurf attack) is used for achieving denial of service attacks when the source address is replaced with a victim's address (spoofing). An attacker with low bandwidth can generate a lot of traffic if he uses good amplifier sites. The spoofed address receives all the ICMP echo replies, creating a denial of service. The amplifier sites are affected in two ways. Some bandwidth wasted and from the victim's point of view they look like an attacker. In the case of Detect 1, the source is not spoofed. The source is systematically scanning all parts of my network to discover subnets that reply. This is the reconnaissance aspect of Detect 1. The scan is fast in time, therefore no attempts to be stealthy. The source is building a database of broadcast addresses with more than one reply and make the information publicly available, hoping that the administrators of those sites will change their configurations to not reply to ICMP broadcasts. The source web site claims that no network goes more than 2 weeks without being rescanned.

Although the intent of the source site is to pressure administrators of vulnerable subnets by exposing them publicly, a side effect is an easy to get, complete list of Smurf-amplifiers for whoever wants to stage an attack. It reminds me of the debate "should a system vulnerability be made public? If yes, then when?

6. Correlations: (Detect 1)

This type of scan to detect Smurf-amplifiers are mentioned on GIAC:

Global Incident Analysis Center - Detects Analyzed 3/5/00 - http://www.sans.org/y2k/030500.htm

7. Evidence of active targeting: (Detect 1)

This is a general scan of my network. No specific host is targeted. This is in accordance with the information posted on the source site, which claim that they scan all networks.

8. Severity: (Detect 1)

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)
-1 = (4 + 2) – (2 + 5)

Criticality = 4 (many of my hosts could be used as Smurf-amplifiers)
Lethality = 2 (if my hosts reply and later used in Smurf attacks, wasted bandwidth, not as bad as victim)
System Countermeasures = 2 (if it would reach individual hosts they would probably reply)
Network Countermeasures = 5 (the network blocks these request)

One aspect to consider is that the price of being used as a Smurf-amplifier may be bigger than the direct effect of wasted bandwidth on my network. My organization would be seen as the attacker from the victim's point of view. For some types of organization, this kind of embarrassment should not be neglected. This would normally be taken into account in the threat and risk assessment process.

9. Defensive recommendation: (Detect 1)

The defensive recommendation is a network configuration that prevents replying to ICMP echo request on broadcast addresses. It is also good practice to block all ICMP echo requests from the outside for most internal hosts, and not only to broadcast addresses. In the case of my network, further analysis showed that not a single reply was generated by the whole scan. The border gateway drops these echo requests.

10. Multiple choice test question (Detect 1)

Based on this trace:

19:52:16.765283 209.221.143.119 > MY.NET.102.255: icmp: echo request (ttl 235, id 40319)
19:52:16.779988 209.221.143.119 > MY.NET.102.0: icmp: echo request (ttl 236, id 40318)
19:52:16.780242 209.221.143.119 > MY.NET.103.0: icmp: echo request (ttl 236, id 40320)
19:52:16.782285 209.221.143.119 > MY.NET.104.0: icmp: echo request (ttl 236, id 40323)
19:52:16.784679 209.221.143.119 > MY.NET.103.255: icmp: echo request (ttl 235, id 40322)
19:52:16.785161 209.221.143.119 > MY.NET.104.255: icmp: echo request (ttl 235, id 40324)

Would you conclude that?

a) the source is scanning for live routers on MY.NET
b) many spoofed addresses of MY.NET are used in a Smurf attack
c) the source is scanning MY.NET to find potential Smurf-amplifiers
d) the source is attempting a large denial of service on MY.NET

Answer: c

Detect 2

05:52:10.555135 P HOST.FOREIGN.NET.3618 > MY.NET.237.251.netbios-ssn: S
24974272:24974272(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 108, id 41158)
05:52:11.457963 P HOST.FOREIGN.NET.3619 > MY.NET.237.252.netbios-ssn: S
24981275:24981275(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 108, id 41926)
05:52:15.514718 P HOST.FOREIGN.NET.3621 > MY.NET.237.253.netbios-ssn: S
24988281:24988281(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 108, id 43718)
05:52:15.578149 P HOST.FOREIGN.NET.3617 > MY.NET.237.250.netbios-ssn: S
24967271:24967271(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 108, id 43974)
05:52:17.487909 P HOST.FOREIGN.NET.3619 > MY.NET.237.252.netbios-ssn: S
24981275:24981275(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 108, id 45254)
05:52:18.498834 P HOST.FOREIGN.NET.3621 > MY.NET.237.253.netbios-ssn: S
24988281:24988281(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 108, id 45510)
05:52:22.513408 P HOST.FOREIGN.NET.3622 > MY.NET.237.254.netbios-ssn: S
24995281:24995281(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 108, id 46278)
05:52:24.527847 P HOST.FOREIGN.NET.3621 > MY.NET.237.253.netbios-ssn: S
24988281:24988281(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 108, id 47302)
05:52:25.435006 P HOST.FOREIGN.NET.3622 > MY.NET.237.254.netbios-ssn: S
24995281:24995281(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 108, id 47558)
05:52:29.568854 P HOST.FOREIGN.NET.3619 > MY.NET.237.252.netbios-ssn: S
24981275:24981275(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 108, id 49862)

05:52:31.492991 P HOST.FOREIGN.NET.3622 > MY.NET.237.254.netbios-ssn: S
24995281:24995281(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 108, id 51142)
05:52:36.618132 P HOST.FOREIGN.NET.3621 > MY.NET.237.253.netbios-ssn: S
24988281:24988281(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 108, id 55750)
05:52:43.563272 P HOST.FOREIGN.NET.3622 > MY.NET.237.254.netbios-ssn: S
24995281:24995281(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 108, id 58822)
06:22:13.891656 P HOST.FOREIGN.NET.3963 > MY.NET.236.1.netbios-ssn: S 26786841:26786841(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 37334)
06:22:16.865780 P HOST.FOREIGN.NET.3963 > MY.NET.236.1.netbios-ssn: S 26786841:26786841(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 38102)
06:22:20.895791 P HOST.FOREIGN.NET.3964 > MY.NET.236.2.netbios-ssn: S 26793846:26793846(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 38870)
06:22:22.896321 P HOST.FOREIGN.NET.3963 > MY.NET.236.1.netbios-ssn: S 26786841:26786841(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 39638)
06:22:23.901621 P HOST.FOREIGN.NET.3964 > MY.NET.236.2.netbios-ssn: S 26793846:26793846(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 40150)
06:22:27.890379 P HOST.FOREIGN.NET.3965 > MY.NET.236.3.netbios-ssn: S 26800856:26800856(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 41174)
06:22:29.951460 P HOST.FOREIGN.NET.3964 > MY.NET.236.2.netbios-ssn: S 26793846:26793846(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 41686)
06:22:30.839528 P HOST.FOREIGN.NET.3965 > MY.NET.236.3.netbios-ssn: S 26800856:26800856(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 41942)
06:22:34.906618 P HOST.FOREIGN.NET.3966 > MY.NET.236.4.netbios-ssn: S 26807861:26807861(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 42966)
06:22:34.961292 P HOST.FOREIGN.NET.3963 > MY.NET.236.1.netbios-ssn: S 26786841:26786841(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 43222)
06:22:36.839760 P HOST.FOREIGN.NET.3965 > MY.NET.236.3.netbios-ssn: S 26800856:26800856(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 43734)
06:22:37.866184 P HOST.FOREIGN.NET.3966 > MY.NET.236.4.netbios-ssn: S 26807861:26807861(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 43990)
06:22:41.906693 P HOST.FOREIGN.NET.3967 > MY.NET.236.5.netbios-ssn: S 26814871:26814871(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 44246)
06:22:41.988783 P HOST.FOREIGN.NET.3964 > MY.NET.236.2.netbios-ssn: S 26793846:26793846(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 44502)
06:22:43.908545 P HOST.FOREIGN.NET.3966 > MY.NET.236.4.netbios-ssn: S 26807861:26807861(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 45270)
06:22:44.892675 P HOST.FOREIGN.NET.3967 > MY.NET.236.5.netbios-ssn: S 26814871:26814871(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 45526)
06:22:48.908741 P HOST.FOREIGN.NET.3968 > MY.NET.236.6.netbios-ssn: S 26821881:26821881(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 46294)
06:22:48.917258 P HOST.FOREIGN.NET.3965 > MY.NET.236.3.netbios-ssn: S 26800856:26800856(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 46550)
06:22:50.937626 P HOST.FOREIGN.NET.3967 > MY.NET.236.5.netbios-ssn: S 26814871:26814871(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 46806)
06:22:51.842506 P HOST.FOREIGN.NET.3968 > MY.NET.236.6.netbios-ssn: S 26821881:26821881(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 47574)
06:22:55.917875 P HOST.FOREIGN.NET.3969 > MY.NET.236.7.netbios-ssn: S 26828891:26828891(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 47830)
06:22:55.978850 P HOST.FOREIGN.NET.3966 > MY.NET.236.4.netbios-ssn: S 26807861:26807861(0)
win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 48086)

06:22:57.867594 P HOST.FOREIGN.NET.3968 > MY.NET.236.6.netbios-ssn: S 26821881:26821881(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 48342)

1. Source of trace (Detect 2)

This traffic is from my network. It is sniffed at the main gateway to the Internet.

2. Detect was generated by: (Detect 2)

Shadow intrusion detection system. This Shadow system (version 1.6) is setup as an experimental facility. It was setup to collect the raw network traffic, to be later used by analysts experimenting with filters. This trace was generated by me using tcpdump filters on the raw traffic data archived by the Shadow system. In this particular case, I was looking for activities on port 139, the NETBIOS session service port, which is known to be used by exploits. I extracted any traffic with destination port of 139, and then noticed this scan on my network. The trace is in a tcpdump output format. Here is a TCP sample with description:

Sample:
05:52:10.555135 P HOST.FOREIGN.NET.3618 > MY.NET.237.251.netbios-ssn: S 24974272:24974272(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 108, id 41158)

Description:
05:52:10.555135 = timestamp (hh:mm:ss.ssssss)
P = this characters always shows up in my output, it is apparently an artifact of the Linux flavor used. It is not a protocol flag (not a PUSH). Not used for the analysis.
HOST.FOREIGN.NET.36118 = source host network address (sanitized) and source port (36118)
MY.NET.237.251.netbios-ssn = destination IP address (sanitized) and destination port (netbios-ssn=139)
S 24974272:24974272(0) = TCP SYN flag set, sequence numbers (begin:end), data size in bytes (0)
win 8192 = window size
<mss 536,nop,nop,sackOK> = TCP options (e.g. mss is maximum segment size)
(DF) = IP Don't Fragment flag
(ttl 108, id 41158) = Time To Live value (108) and Packet Identification number (41158)

3. Probability the source address was spoofed (Detect 2)

It is unlikely that the source address is spoofed because this trace looks like a reconnaissance scan, and the source needs to get the reply in order to gather information about the hosts on my network. A whois query pointed at a Chinese telecom company.

4. Description of attack: (Detect 2)

Scans of hosts on my network to find out if there are hosts listening on the NETBIOS session service port (139). There is a known exploit for this port. On Windows NT/2000 computers, users or shares can be detected using a null session on this port.

Reference: Common Vulnerabilities and Exposures

CAN-1999-0519	** CANDIDATE (under review) ** A NETBIOS/SMB share password is the default, null, or missing.

5. Attack mechanism: (Detect 2)

This attack is a scan to discover hosts that are vulnerable to "null session" attacks. The source host is systematically scanning each host on my network by trying to establish a TCP connection with a SYN. The source attempts to connect to each destination host 3 times, which could be a normal TCP retry behavior. The source port is incremented with each destination host, which would happen if the scanning tool uses system calls. The pace of the scan is 34 SYNs per minute (i.e. 11 hosts/min.). This is relatively slow and could be a way to stay below a certain IDS alert threshold.

If a destination hosts replied to those SYNs it would become a potential target for "null session" attacks. A null session attacks. A null session is a NetBIOS connection established with a zero length string as user, password, and domain name, which is designed to enable enumeration of shares and users. It was also found to allow access to the registry with the same level of permissions as the Everyone group

6. Correlations: (Detect 2)

There are many references to port 139 exploitation from personal firewall developers. Some scans on this port were reported at: http://www.physiology.rwth-aachen.de/~jens/wwp.html#t10

7. Evidence of active targeting: (Detect 2)

This is a general scan of my network. No specific host is targeted. The full list of destination hosts includes many inactive IP numbers. The source does not seem to know a priori which of my hosts are Windows boxes as it scans all of them.

8. Severity: (Detect 2)

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)
-1 = (3 + 3) – (2 + 5)

Criticality = 3 (my whole network is targeted)
Lethality = 3 (valuable information could be revealed in successful null session)
System Countermeasures = 2
Network Countermeasures = 5 (the border gateway blocks these port from the Internet)

9. Defensive recommendation: (Detect 2)

An investigation of the complete trace showed that no hosts replied the TCP connection attempts. It is good practice to block access to the NETBIOS ports (135-139) from outside a corporate network, and to verify that they are actually blocked. On a small home network, if one must share files, it is safer to use NetBEUI or IPX to transfer the files rather than TCP/IP.

10. Multiple choice test question (Detect 2)

06:22:36.839760 HOST.FOREIGN.NET.3965 > MY.NET.236.3.139: S 26800856:26800856(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 43734)
06:22:37.866184 HOST.FOREIGN.NET.3966 > MY.NET.236.4.139: S 26807861:26807861(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 43990)
06:22:41.906693 HOST.FOREIGN.NET.3967 > MY.NET.236.5.139: S 26814871:26814871(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 44246)

06:22:41.988783 HOST.FOREIGN.NET.3964 > MY.NET.236.2.139: S 26793846:26793846(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 44502)
06:22:43.908545 HOST.FOREIGN.NET.3966 > MY.NET.236.4.139: S 26807861:26807861(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 45270)
06:22:44.892675 HOST.FOREIGN.NET.3967 > MY.NET.236.5.139: S 26814871:26814871(0) win 8192 <mss 536,nop,nop,sackOK> (DF) (ttl 109, id 45526)

What is HOST.FOREIGN.NET trying to do?

a) load balancing
b) discover potential targets for null session exploits
c) denial of service
d) finding which source port is suitable for destination port 139

Answer: b

Detect 3

10:41:07.151930 P HOST.BIGNAME.EDU.sunrpc > MY.NET.10.14.sunrpc: SF 1405366985:1405366985(0) win 1028 (ttl 31, id 39426)
10:41:07.151997 P HOST.BIGNAME.EDU.sunrpc > MY.NET.10.15.sunrpc: SF 1817927555:1817927555(0) win 1028 (ttl 31, id 39426)
10:41:07.155512 P HOST.BIGNAME.EDU.sunrpc > MY.NET.10.22.sunrpc: SF 1677527703:1677527703(0) win 1028 (ttl 31, id 39426)
10:41:07.155580 P HOST.BIGNAME.EDU.sunrpc > MY.NET.10.25.sunrpc: SF 607499012:607499012(0) win 1028 (ttl 31, id 39426)
10:41:07.159133 P HOST.BIGNAME.EDU.sunrpc > MY.NET.10.42.sunrpc: SF 473126440:473126440(0) win 1028 (ttl 31, id 39426)
10:41:07.159162 P HOST.BIGNAME.EDU.sunrpc > MY.NET.10.52.sunrpc: SF 1927185356:1927185356(0) win 1028 (ttl 31, id 39426)
10:41:07.159229 P HOST.BIGNAME.EDU.sunrpc > MY.NET.10.53.sunrpc: SF 1167671860:1167671860(0) win 1028 (ttl 31, id 39426)
10:41:07.161440 P HOST.BIGNAME.EDU.sunrpc > MY.NET.10.61.sunrpc: SF 745606514:745606514(0) win 1028 (ttl 31, id 39426)
10:41:07.163089 P HOST.BIGNAME.EDU.sunrpc > MY.NET.10.62.sunrpc: SF 164504862:164504862(0) win 1028 (ttl 31, id 39426)
10:41:07.166656 P HOST.BIGNAME.EDU.sunrpc > MY.NET.10.66.sunrpc: SF 1585136521:1585136521(0) win 1028 (ttl 31, id 39426)
10:41:07.166723 P HOST.BIGNAME.EDU.sunrpc > MY.NET.10.71.sunrpc: SF 60654602:60654602(0) win 1028 (ttl 31, id 39426)
10:41:07.170241 P HOST.BIGNAME.EDU.sunrpc > MY.NET.10.73.sunrpc: SF 1323437936:1323437936(0) win 1028 (ttl 31, id 39426)
10:41:07.170307 P HOST.BIGNAME.EDU.sunrpc > MY.NET.10.74.sunrpc: SF 1275247030:1275247030(0) win 1028 (ttl 31, id 39426)
10:41:07.173834 P HOST.BIGNAME.EDU.sunrpc > MY.NET.10.94.sunrpc: SF 1221002675:1221002675(0) win 1028 (ttl 31, id 39426)

1. Source of trace (Detect 3)

This traffic is from my network. It is sniffed at the main gateway to the Internet.

2. Detect was generated by: (Detect 3)

Shadow intrusion detection system. This Shadow system (version 1.6) is setup as an experimental facility. It was setup to collect the raw network traffic, to be later used by analysts experimenting with filters. This trace was generated by me, using tcpdump filters on the raw traffic data archived by the Shadow system. In this case, I was looking for TCP flag settings typical of traffic trying to be stealthy, in particular the SYN/FIN combination. I noticed a SYN/FIN scan from a particular host and filtered to isolate it. The tcpdump filter looks like this:

```
ip and tcp and (tcp[13] & 0x03 =3) and host HOST.BIGNAME.EDU
```

(See Detect 2 for output format description)

3. Probability the source address was spoofed (Detect 3)

It is unlikely that the source address is spoofed because this trace is a reconnaissance scan, and the source needs to get the reply in order to gather information about the hosts on my network. The source IP address belongs to a prestigious university in the USA. University computers are sometime compromised by outsiders to and used for malicious activity.

4. Description of attack: (Detect 3)

This is a SYN/FIN host scan on port 111 (sunrpc). The attacker is looking for Remote Procedure Call (RPC) services running on my hosts to later exploit some of the vulnerabilities known for the RPC services. An example would be the rpc.cmsd overflow exploit, which could be used to execute arbitrary commands.

Reference: Common Vulnerabilities and Exposures

CVE-1999-0696	Buffer overflow in CDE Calendar Manager Service Daemon (rpc.cmsd)

5. Attack mechanism: (Detect 3)

This attack is a scan to discover Unix hosts with RPC programs waiting for connections. The connections to the port 111, if successful, would be answered by rpcbind or portmapper. The reply would then indicate the RPC services running and the attacker would use that information to select his favorite RPC service exploit.

Let's describe the scan trace in more details. The pace of the scan is fast in time. Both the SYN and the FIN TCP flags are set, which is not normal. This is a know technique to evade some packet filtering devices which may, for instance, block only access to port 111 for a normal SYN attempt.

There are two other interesting points about the trace. First, the source port is the same as the destination port (111). This is not really required to access the portmapper. Second, the packet ID is the same for all packets (39426). This is probably a signature of the attack tool. Correlation research on GIAC reports shows SYN/FIN port scans with the exact same packet ID number (39426) and the destination port number equal to the source port number. Examples of these traces are at:

http://www.sans.org/y2k/070200-2000.htm

http://www.sans.org/y2k/050200.htm

6. Correlations: (Detect 3)

This GIAC reference contains a similar SYN/FIN scan (to DNS port), with the same tool signature and also a port 111 scan.

http://www.sans.org/y2k/032000.htm

7. Evidence of active targeting: (Detect 3)

This is a general scan of my network. No specific host is targeted. Although the section of the trace posted in this report may show that some hosts number are missing, the full trace shows that all numbers are present but not in a clean sequence. The source does not seem to know a priori which of my hosts are Sun Solaris boxes (sunrpc port) as it scans all of them.

8. Severity: (Detect 3)

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)
0 = (3 + 4) – (2 + 5)

Criticality = 3 (my whole network is targeted by a scan)
Lethality = 4 (RPC services exploit could be serious)
System Countermeasures = 3
Network Countermeasures = 5 (the border gateway blocks these port from the Internet)

9. Defensive recommendation: (Detect 3)

Block access from Internet to port 111 whenever possible regardless of the TCP flags set, and verify that it works. Some of the RPC services may required, so make sure that all the relevant patches are installed. Also, the reply which lists the services (rpcinfo) may be disabled.

10. Multiple choice test question (Detect 3)

10:41:07.161440 HOST.BIGNAME.EDU.sunrpc > MY.NET.10.61.sunrpc: SF 745606514:745606514(0) win 1028 (ttl 31, id 39426)
10:41:07.163089 HOST.BIGNAME.EDU.sunrpc > MY.NET.10.62.sunrpc: SF 164504862:164504862(0) win 1028 (ttl 31, id 39426)
10:41:07.166656 HOST.BIGNAME.EDU.sunrpc > MY.NET.10.66.sunrpc: SF 1585136521:1585136521(0) win 1028 (ttl 31, id 39426)
10:41:07.166723 HOST.BIGNAME.EDU.sunrpc > MY.NET.10.71.sunrpc: SF 60654602:60654602(0) win 1028 (ttl 31, id 39426)
10:41:07.170241 HOST.BIGNAME.EDU.sunrpc > MY.NET.10.73.sunrpc: SF 1323437936:1323437936(0) win 1028 (ttl 31, id 39426)
10:41:07.170307 HOST.BIGNAME.EDU.sunrpc > MY.NET.10.74.sunrpc: SF 1275247030:1275247030(0) win 1028 (ttl 31, id 39426)

In this trace, what could be a sign that an attack tool generated it?

a) SYN and FIN are both set
b) the packet ID is the same for all packets

c) the source port and the destination port are both sunrpc (111)
d) all of the above

answer: d

Detect 4

18:52:31.264953 P 64.71.131.44.13269 > MY.NET.135.206.1163: R 0:0(0) ack 674719802 win 0 (ttl 55, id 59623)
18:52:32.104755 P 64.71.131.44.17022 > MY.NET.175.133.1163: R 0:0(0) ack 674719802 win 0 (ttl 55, id 61164)
18:52:33.864967 P 64.71.131.44.25101 > MY.NET.153.8.1163: R 0:0(0) ack 674719802 win 0 (ttl 55, id 26692)
18:52:37.066629 P 64.71.131.44.2654 > MY.NET.95.133.1243: R 0:0(0) ack 674719802 win 0 (ttl 53, id 53833)
18:52:40.710212 P 64.71.131.44.3101 > MY.NET.62.233.2212: R 0:0(0) ack 674719802 win 0 (ttl 53, id 9151)
18:52:43.246199 P 64.71.131.44.3101 > MY.NET.62.233.2212: R 0:0(0) ack 1 win 0 (ttl 53, id 45912)
18:52:53.443359 P 64.71.131.44.4184 > MY.NET.205.4.1412: R 0:0(0) ack 674719802 win 0 (ttl 53, id 41721)
18:52:54.988496 P 64.71.131.44.45374 > MY.NET.249.159.1163: R 0:0(0) ack 674719802 win 0 (ttl 53, id 64715)
18:52:57.152022 P 64.71.131.44.2117 > MY.NET.10.108.1403: R 0:0(0) ack 674719802 win 0 (ttl 53, id 18620)
18:53:09.399683 P 64.71.131.44.5496 > MY.NET.108.13.2292: R 0:0(0) ack 674719802 win 0 (ttl 53, id 58244)
18:53:09.568816 P 64.71.131.44.5496 > MY.NET.108.13.2292: R 0:0(0) ack 1 win 0 (ttl 53, id 26360)
18:53:10.257344 P 64.71.131.44.8201 > MY.NET.157.237.1591: R 0:0(0) ack 674719802 win 0 (ttl 53, id 8641)
18:53:28.648904 P 64.71.131.44.8590 > MY.NET.65.95.2292: R 0:0(0) ack 674719802 win 0 (ttl 53, id 64870)
18:53:54.990773 P 64.71.131.44.17867 > MY.NET.10.189.2212: R 0:0(0) ack 674719802 win 0 (ttl 55, id 59093)
18:54:04.146890 P 64.71.131.44.9597 > MY.NET.26.123.1742: R 0:0(0) ack 674719802 win 0 (ttl 55, id 47503)
18:54:07.291858 P 64.71.131.44.17867 > MY.NET.10.189.2212: R 0:0(0) ack 1 win 0 (ttl 55, id 4531)
18:54:24.480494 P 64.71.131.44.15473 > MY.NET.206.96.1403: R 0:0(0) ack 674719802 win 0 (ttl 53, id 5250)
18:54:26.398717 P 64.71.131.44.23082 > MY.NET.156.208.1153: R 0:0(0) ack 674719802 win 0 (ttl 53, id 47658)
18:54:28.738882 P 64.71.131.44.24329 > MY.NET.91.87.1153: R 0:0(0) ack 674719802 win 0 (ttl 55, id 54135)
18:54:29.956284 P 64.71.131.44.14644 > MY.NET.35.32.2193: R 0:0(0) ack 674719802 win 0 (ttl 53, id 48706)
18:54:30.162606 P 64.71.131.44.14644 > MY.NET.35.32.2193: R 0:0(0) ack 1 win 0 (ttl 53, id 1739)
18:54:30.519873 P 64.71.131.44.14644 > MY.NET.35.32.2193: R 0:0(0) ack 1 win 0 (ttl 53, id 10941)
18:54:32.077821 P 64.71.131.44.19685 > MY.NET.219.195.1502: R 0:0(0) ack 674719802 win 0 (ttl 53, id 6106)
18:54:32.902430 P 64.71.131.44.22964 > MY.NET.125.197.2212: R 0:0(0) ack 674719802 win 0 (ttl 53, id 26816)
18:54:34.352097 P 64.71.131.44.60026 > MY.NET.183.1.1163: R 0:0(0) ack 674719802 win 0 (ttl 55, id 60258)
18:54:38.404614 P 64.71.131.44.20954 > MY.NET.86.89.1502: R 0:0(0) ack 674719802 win 0 (ttl 55, id 53078)
18:54:38.948925 P 64.71.131.44.22964 > MY.NET.125.197.2212: R 0:0(0) ack 1 win 0 (ttl 53, id 27757)
18:54:56.704594 P 64.71.131.44.21550 > MY.NET.143.32.2292: R 0:0(0) ack 674719802 win 0 (ttl 53, id 149)
18:55:03.328968 P 64.71.131.44.23018 > MY.NET.1.94.1323: R 0:0(0) ack 674719802 win 0 (ttl 53, id 52461)
18:55:17.047121 P 64.71.131.44.29243 > MY.NET.38.37.2212: R 0:0(0) ack 674719802 win 0 (ttl 53, id 65405)
18:55:19.161270 P 64.71.131.44.27808 > MY.NET.59.182.1502: R 0:0(0) ack 674719802 win 0 (ttl 53, id 55066)
18:55:21.894106 P 64.71.131.44.24679 > MY.NET.236.214.1671: R 0:0(0) ack 674719802 win 0 (ttl 55, id 24394)
18:55:22.309434 P 64.71.131.44.25713 > MY.NET.176.117.1323: R 0:0(0) ack 674719802 win 0 (ttl 55, id 30602)
18:55:24.749258 P 64.71.131.44.29243 > MY.NET.38.37.2212: R 0:0(0) ack 1 win 0 (ttl 53, id 27484)
18:55:38.078357 P 64.71.131.44.34865 > MY.NET.203.252.1153: R 0:0(0) ack 674719802 win 0 (ttl 55, id 1051)
18:55:44.459454 P 64.71.131.44.24115 > MY.NET.208.143.1652: R 0:0(0) ack 674719802 win 0 (ttl 53, id 53868)
18:55:44.603734 P 64.71.131.44.24115 > MY.NET.208.143.1652: R 0:0(0) ack 1 win 0 (ttl 53, id 44609)
18:55:50.177750 P 64.71.131.44.24763 > MY.NET.170.53.1742: R 0:0(0) ack 674719802 win 0 (ttl 53, id 39675)
18:56:06.566824 P 64.71.131.44.33404 > MY.NET.100.247.1412: R 0:0(0) ack 674719802 win 0 (ttl 55, id 18219)
18:56:11.116097 P 64.71.131.44.29380 > MY.NET.26.167.1483: R 0:0(0) ack 674719802 win 0 (ttl 53, id 12314)
18:56:17.429423 P 64.71.131.44.29236 > MY.NET.189.102.1041: R 0:0(0) ack 674719802 win 0 (ttl 55, id 50827)
18:56:18.020523 P 64.71.131.44.30652 > MY.NET.139.137.2193: R 0:0(0) ack 674719802 win 0 (ttl 55, id 40819)
18:56:18.105743 P 64.71.131.44.30652 > MY.NET.139.137.2193: R 0:0(0) ack 1 win 0 (ttl 55, id 45681)
18:56:18.338424 P 64.71.131.44.30652 > MY.NET.139.137.2193: R 0:0(0) ack 1 win 0 (ttl 55, id 46022)

1. Source of trace (Detect 4)

This traffic is from my network. It is sniffed at the main gateway to the Internet.

2. Detect was generated by: (Detect 4)

Shadow intrusion detection system. This Shadow system (version 1.6) is setup as an experimental facility. It was setup to collect the raw network traffic, to be later used by analysts experimenting with filters. This trace was generated by me, using tcpdump filters on the raw traffic data archived by the Shadow system.

3. Probability the source address was spoofed (Detect 4)

It is unlikely that the source address is spoofed because this trace is a reconnaissance scan, and the source needs to get the reply in order to gather information about the hosts on my network.

4. Description of attack: (Detect 4)

This is a RESET scan. It is meant to provoke responses that could be used to map my network. Some routers may return information on the host targeted not being reachable

5. Attack mechanism: (Detect 4)

This is some sort of stealthy technique to map networks. The scanner tries to hit destination host with a TCP RESET packet, without prior exchange with that host. From the destination host point of view it is a response to a stimulus that it did not sent. The scanner may get replies from routers saying that certain hosts are not reachable because the router may assume that the inside hosts did initiate the connection with the host. In fact, the router does not know because it does not keep track of the states of connections. The scanner concludes that the hosts for which the unreachable was not sent really exists. This is called inverse mapping.

6. Correlations: (Detect 4)

I found a similar trace on page in the book *Network Intrusion Detection – An analyst's handbook*, by Stephen Northcutt (NewRiders Publishing). The trace in the book is probably from the same attack tool because the packet sequence number is constant and identical to the one in my trace (674719802).

7. Evidence of active targeting: (Detect 4)

This is a general scan of my network. No specific host seems targeted.

8. Severity: (Detect 4)

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)
-2 = (3 + 2) – (2 + 5)

Criticality = 3 (many of my hosts could be used as Smurf-amplifiers)
Lethality = 2 (only scanning, for now)
System Countermeasures = 2
Network Countermeasures = 5 (the network blocks these request)

9. Defensive recommendation: (Detect 4)

Defenses are fine, our border router did not return any information to the scanning source during the whole scan.

10. Multiple choice test question (Detect 4)

```
18:52:31.264953 P 64.71.131.44.13269 > MY.NET.135.206.1163: R 0:0(0) ack 674719802 win 0 (ttl 55, id 59623)
18:52:32.104755 P 64.71.131.44.17022 > MY.NET.175.133.1163: R 0:0(0) ack 674719802 win 0 (ttl 55, id 61164)
18:52:33.864967 P 64.71.131.44.25101 > MY.NET.153.8.1163: R 0:0(0) ack 674719802 win 0 (ttl 55, id 26692)
18:52:37.066629 P 64.71.131.44.2654 > MY.NET.95.133.1243: R 0:0(0) ack 674719802 win 0 (ttl 53, id 53833)
18:52:40.710212 P 64.71.131.44.3101 > MY.NET.62.233.2212: R 0:0(0) ack 674719802 win 0 (ttl 53, id 9151)
18:52:43.246199 P 64.71.131.44.3101 > MY.NET.62.233.2212: R 0:0(0) ack 1 win 0 (ttl 53, id 45912)
18:52:53.443359 P 64.71.131.44.4184 > MY.NET.205.4.1412: R 0:0(0) ack 674719802 win 0 (ttl 53, id 41721)
18:52:54.988496 P 64.71.131.44.45374 > MY.NET.249.159.1163: R 0:0(0) ack 674719802 win 0 (ttl 53, id 64715)
18:52:57.152022 P 64.71.131.44.2117 > MY.NET.10.108.1403: R 0:0(0) ack 674719802 win 0 (ttl 53, id 18620)
18:53:09.399683 P 64.71.131.44.5496 > MY.NET.108.13.2292: R 0:0(0) ack 674719802 win 0 (ttl 53, id 58244)
18:53:09.568816 P 64.71.131.44.5496 > MY.NET.108.13.2292: R 0:0(0) ack 1 win 0 (ttl 53, id 26360)
18:53:10.257344 P 64.71.131.44.8201 > MY.NET.157.237.1591: R 0:0(0) ack 674719802 win 0 (ttl 53, id 8641)
18:53:28.648904 P 64.71.131.44.8590 > MY.NET.65.95.2292: R 0:0(0) ack 674719802 win 0 (ttl 53, id 64870)
18:53:54.990773 P 64.71.131.44.17867 > MY.NET.10.189.2212: R 0:0(0) ack 674719802 win 0 (ttl 55, id 59093)
18:54:04.146890 P 64.71.131.44.9597 > MY.NET.26.123.1742: R 0:0(0) ack 674719802 win 0 (ttl 55, id 47503)
18:54:07.291858 P 64.71.131.44.17867 > MY.NET.10.189.2212: R 0:0(0) ack 1 win 0 (ttl 55, id 4531)
```

What do you think this trace is?

a) MY.NET hosts are replying to a Smurf attack
b) 64.71.131.44 is a defective router
c) a traceroute
d) a RESET scan for mapping purposes

answer: d

Assignment 2 – Evaluate and Attack

Attack name: Teardrop
Attack type: Denial of service
Attack tool: Targa 2

The attack tool is Targa 2 (http://mixter.warrior2k.com/targa2.c). The author is Mixter, also known as the author of TFN, a distributed denial of service tool. Targa 2 is an interface to eleven multi-platform remote denial of service (DoS) exploits. Here is the header of the source program that includes the list of DoS exploits, their author, and the vulnerable platforms.

```
/* targa2.c - copyright by Mixter <mixter@popmail.com>
   version 2.1 - released 22/3/99 - interface to 11
   multi-platform remote denial of service exploits
```

```
  gcc -Wall -O2 targa2.c -o targa2 ; strip targa2 */

/*
 *       featured exploits / authors / vulnerable platforms
 * bonk by route|daemon9 & klepto            - win95, nameservers
 * jolt by Jeff W. Roberson (overdrop: Mixter) - win95, klog (old linux)
 * land by m3lt                        - win95/nt, old un*x's
 * nestea by humble & ttol                - older linux/bsd?
 * newtear by route|daemon9                - linux/bsd/win95/others
 * syndrop by PineKoan                - linux/win95/?
 * teardrop by route|daemon9                - lots of os's
 * winnuke by _eci                    - win95/win31
 * 1234 by DarkShadow/Flu                - win95/98/nt/others?
 * saihyousen by noc-wage                - win98/firewalls/routers
 * oshare by r00t zer0                - win9x/NT/macintosh
 */

/* http://members.xoom.com/i0wnu - code copyright by Mixter */
```

Teardrop Attack

Teardrop is an attack that aims at crashing vulnerable systems by sending them fragmented packets with illegal offset values that cause the destination system to reassemble the data with an overlap. Some systems were found to "badly react" to this. Here is Common Vulnerabilities and Exposures (CVE) entry and a Microsoft Support Article about this vulnerability.

CVE version: 20000712	
Name	**Description**
CAN-1999-0015	** CANDIDATE (under review) ** Teardrop IP denial of service.

Microsoft Knowledge Base Article - Q154174

Invalid ICMP Datagram Fragments Hang Windows NT, Windows 95

The information in this article applies to:

- Microsoft Windows NT Workstation versions 3.51, 4.0
- Microsoft Windows NT Server versions 3.51, 4.0
- Microsoft Windows 95
- Microsoft Windows NT Server version 4.0, Terminal Server Edition

SYMPTOMS

Computers running Windows NT or Windows 95 may stop responding (hang) when they receive corrupted Internet Control Message Protocol (ICMP) datagram fragments from a client.

RESOLUTION

Microsoft has updated the TCP/IP protocol stack to correct this problem. Instructions for installing it are available from Microsoft support channels or directly from the Internet locations below.

…

Targa was installed on one host on a small lab network and was used to generate many simultaneous DoS attacks (Teardrop and others) on the same LAN. Here is a Targa Teardrop attack trace for one of the source address, followed by a detailed explanation of the trace.

```
23:57:27.167915 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 0 win 512 urg 0 (frag 242:40@0+)
23:57:27.168001 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
23:57:27.187773 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 1 win 512 urg 0 (frag 242:40@0+)
23:57:27.187860 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
23:57:27.207773 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 1 win 512 urg 0 (frag 242:40@0+)
23:57:27.207858 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
23:57:27.227787 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 1 win 512 urg 0 (frag 242:40@0+)
23:57:27.227873 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
23:57:27.247780 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 1 win 512 urg 0 (frag 242:40@0+)
23:57:27.247864 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
23:57:27.267921 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 1 win 512 urg 0 (frag 242:40@0+)
23:57:27.268007 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
23:57:27.287773 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 1 win 512 urg 0 (frag 242:40@0+)
23:57:27.287857 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
23:57:27.307760 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 1 win 512 urg 0 (frag 242:40@0+)
23:57:27.307846 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
23:57:27.327847 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 1 win 512 urg 0 (frag 242:40@0+)
23:57:27.327930 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
23:57:27.347764 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 1 win 512 urg 0 (frag 242:40@0+)
23:57:27.347848 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
23:57:27.367743 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 1 win 512 urg 0 (frag 242:40@0+)
23:57:27.367828 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
23:57:27.387780 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 1 win 512 urg 0 (frag 242:40@0+)
23:57:27.387864 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
23:57:27.407780 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 1 win 512 urg 0 (frag 242:40@0+)
23:57:27.407864 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
23:57:27.427848 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 1 win 512 urg 0 (frag 242:40@0+)
23:57:27.427932 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
23:57:27.447792 P 160.247.118.26.35992 > 172.30.0.229.51934: . 0:40(40) ack 1 win 512 urg 0 (frag 242:40@0+)
23:57:27.447865 P 160.247.118.26 > 172.30.0.229: (frag 242:4@24)
```

Explanation:

The first two packets represent a complete Teardrop attack. The first packet is a fragmented packet with these characteristics:

- TCP
- fragment ID = 242
- datagram length = 40 bytes
- fragment offset = 0 (the + means more fragments to come)

The second fragment has these characteristics:

- fragment ID = 242 (same as first fragment, therefore it is the second fragment)
- datagram length = 4 bytes
- fragment offset = **24** (no + means this is the last fragment)

The fragment offset of 24 is not normal. If the first fragment started at offset 0 and had a length of 40 bytes, then the next fragment should have an offset of 40, and not 24. The destination source TCP/IP stack would overlap the second fragment with the first fragment, as shown in this sketch:

```
Fragment 1          0--------------------40
Fragment 2                    24++++

After reassembly    0-----------++++-----40
```

Reassembling with an overlap is known to crash/reboot some versions of some operating systems, which constitutes a denial of service for the user of that system.

One of the signatures of the Teardrop attack is the fragment ID of 242. Some intrusion detection systems used that value in their detection of this attack, but they had to be patched when variants of Teardrop started to change that value. The length and offset values in this trace are also signatures of Teardrop.

The Targa delivery of the Teardrop attack repeated that process 15 times, as fast as it could, with the same spoofed address. I determined that the source address was spoofed because I ran the attack within an isolated LAN, for which the IP addresses are known.

The Targa program was also generating other DoS attacks on the same host, and one of these attacks is relevant to the discussion on Teardrop. A tcpdump output of this other attack is presented below:

```
23:57:07.463870 P 58.83.99.98 > 172.30.0.229: icmp: echo request (frag 4321:380@0+)
23:57:07.464151 P 58.83.99.98 > 172.30.0.229: (frag 4321:380@376+)
23:57:07.464427 P 58.83.99.98 > 172.30.0.229: (frag 4321:380@760+)
23:57:07.464704 P 58.83.99.98 > 172.30.0.229: (frag 4321:380@1136+)
23:57:07.464980 P 58.83.99.98 > 172.30.0.229: (frag 4321:380@1520+)
.
. (packets removed)
.
23:57:07.512401 P 58.83.99.98 > 172.30.0.229: (frag 4321:380@64216+)
23:57:07.512678 P 58.83.99.98 > 172.30.0.229: (frag 4321:380@64600+)
23:57:07.512954 P 58.83.99.98 > 172.30.0.229: (frag 4321:380@64976+)
23:57:07.513230 P 58.83.99.98 > 172.30.0.229: (frag 4321:380@65360)
```

This is a trace of the Targa's Jolt attack. It is some sort of combination of the Teardrop attack, because of the fragment offset anomaly just discussed (376 < 380), and of the Ping O'Death attack, which causes reassembled IP datagrams to be larger than the maximum IP data size (65360 + 380 > 65535).

The best defense for the Teardrop attack is to apply the patches to the TCP/IP stacks of the affected operating systems.

Assignment 3 – "Analyze This" Scenario

(Note: The format is a report from Marc Grégoire to the owner of the MY.NET network)

Introduction

This is a security analysis of your network (MY.NET). Your network was monitored with the Snort intrusion detection system (IDS) over about 40 days. The Snort reports were analyzed to assess the security of the network. This report covers the following:

- Data collection,
- Overall statistics of Snort alert reports,
- Overall statistics of Snort scan reports,
- Detailed analysis of specific alerts and specific hosts activities,
  - Alerts from outside hosts
  - Alerts from inside hosts
  - Port scans from outside hosts
  - Port scans from inside hosts
- Summary and recommendations.

Data Collection

Network activity was monitored with the Snort IDS (http://www.snort.org). Snort is a free IDS used extensively by the network security community. The network was monitored from June 27th to August 10th 2000. There were some gaps in the data collected, but the time periods covered are enough to do a first level assessment of the network activity from a security point of view.

The Snort reports produced are of two types: alert reports and scan reports. The statistics and the analysis presented below tend to treat the two types of reports separately. A large amount of Snort data was collected: 80K+ of alerts and 280K+ of scan events. These reports were generated using a fairly standard set of Snort rules. Each Snort rule is a filter designed to detect a specific type of network traffic that is normally recognized as malicious, also called attack. As with all other IDS, it is common to have these rules triggering on legitimate network traffic. These are called *false positives*, and are part of daily work of network intrusion detection analysts. The were false positives in the Snort reports on the MY.NET network.

Overall statistics of Snort alert reports

This section presents overall statistics produced by processing the complete set of Snort alert reports. It is meant to give a top view of general suspicious activity on your network. Specific events will be discussed in the detailed section.

The distribution of attacks methods for the whole monitoring period is presented in table 1.

Table 1: The distribution of attack methods

%	# of attacks	methods
37.42	30112	Port Scan related (*often more than 1 alert per scan*)
24.94	20067	SYN-FIN scan!
17.36	13972	Watchlist 000220 IL-ISDNNET-990517
5.93	4776	Watchlist 000222 NET-NCFC
4.09	3291	WinGate 8080 Attempt
2.88	2318	Attempted Sun RPC high port access
2.78	2240	WinGate 1080 Attempt
1.48	1188	SNMP public access
1.45	1170	IDS247 - MISC - Large UDP Packet
0.40	323	Napster 8888 Data
0.30	240	SMB Name Wildcard
0.26	206	GIAC 000218 VA-CIRT port 34555
0.23	186	GIAC 000218 VA-CIRT port 35555
0.21	170	Napster 7777 Data
0.12	98	Null scan!
0.06	46	NMAP TCP ping!
0.02	20	SUNRPC highport access!
0.01	12	Napster Client Data
0.01	11	Queso fingerprint
0.01	8	External RPC call
0.01	7	IDS127 - TELNET - Login Incorrect
0.01	5	site exec - Possible wu-ftpd exploit - GIAC000623
0.00	4	Happy 99 Virus
0.00	2	Possible wu-ftpd exploit - GIAC000623
0.00	1	Probable NMAP fingerprint attempt
0.00	1	IDS08 - TELNET - daemon-active
0.00	1	FTP-bad-login
0.00	1	Back Orifice

This table clearly shows that a lot of suspicious activity is taking place on this network. The large number of attacks are not necessarily out of the ordinary for a network of this size, but there are security issues that need to be addressed in the short term and in the long term. Many of the attack methods will be addressed in the detailed analysis section. It is worth mentioning that the number of attacks reported for the "port scan" method is much larger than the number of "complete"port scan detected. This is caused by the way the Snort system reports these in its Alert Report. Further analysis showed that there were 1487 "complete" port scans.

The number of attacks from same host to same destination using the same method are presented in Table 2. Only the top part of the table is included for clarity (the complete table is available from the author). Note also that the port scan attacks are not part of this table.

Table 2: The number of attack from same host to same destination using same method

# of attacks	from	to	with
4323	212.179.38.141	MY.NET.217.38	Watchlist 000220 IL-ISDNNET-990517
3231	212.179.19.134	MY.NET.217.114	Watchlist 000220 IL-ISDNNET-990517
2240	205.188.153.111	MY.NET.217.126	Attempted Sun RPC high port access
1971	212.179.41.218	MY.NET.217.114	Watchlist 000220 IL-ISDNNET-990517
1808	212.179.54.69	MY.NET.182.94	Watchlist 000220 IL-ISDNNET-990517

1702	212.179.23.4	MY.NET.179.51	Watchlist 000220 IL-ISDNNET-990517
1345	159.226.115.1	MY.NET.253.41	Watchlist 000222 NET-NCFC
1183	128.231.171.123	MY.NET.253.105	WinGate 8080 Attempt
1170	211.40.176.214	MY.NET.98.179	IDS247 - MISC - Large UDP Packet
730	212.179.4.238	MY.NET.53.28	Watchlist 000220 IL-ISDNNET-990517
589	159.226.64.164	MY.NET.253.41	Watchlist 000222 NET-NCFC
573	159.226.64.164	MY.NET.253.43	Watchlist 000222 NET-NCFC
546	159.226.64.164	MY.NET.253.42	Watchlist 000222 NET-NCFC
282	24.3.26.53	MY.NET.253.105	WinGate 8080 Attempt
274	159.226.5.77	MY.NET.100.230	Watchlist 000222 NET-NCFC
259	159.226.91.37	MY.NET.100.230	Watchlist 000222 NET-NCFC
242	MY.NET.97.186	MY.NET.101.192	SNMP public access
228	MY.NET.101.160	MY.NET.101.192	SMB Name Wildcard
223	216.0.124.26	MY.NET.253.105	WinGate 8080 Attempt
209	24.3.42.201	MY.NET.253.105	WinGate 8080 Attempt
205	208.184.216.189	MY.NET.201.2	Napster 8888 Data
159	MY.NET.97.80	MY.NET.101.192	SNMP public access
131	MY.NET.97.237	MY.NET.101.192	SNMP public access
110	MY.NET.98.152	MY.NET.101.192	SNMP public access
106	159.226.45.3	MY.NET.253.41	Watchlist 000222 NET-NCFC
90	208.184.216.183	MY.NET.97.204	Napster 7777 Data
87	209.122.205.2	MY.NET.253.105	WinGate 8080 Attempt
85	159.226.45.108	MY.NET.6.7	Watchlist 000222 NET-NCFC
85	212.179.101.218	MY.NET.181.88	Watchlist 000220 IL-ISDNNET-990517
83	24.6.132.179	MY.NET.253.105	WinGate 8080 Attempt
79	MY.NET.97.219	MY.NET.101.192	SNMP public access
78	24.40.24.9	MY.NET.253.105	WinGate 8080 Attempt
75	159.226.5.152	MY.NET.100.165	Watchlist 000222 NET-NCFC
75	MY.NET.97.239	MY.NET.101.192	SNMP public access
72	212.38.168.183	MY.NET.97.101	WinGate 8080 Attempt
70	159.226.45.3	MY.NET.253.43	Watchlist 000222 NET-NCFC
67	MY.NET.97.87	MY.NET.101.192	SNMP public access
64	212.179.123.13	MY.NET.151.33	Watchlist 000220 IL-ISDNNET-990517
64	152.163.224.100	MY.NET.253.24	GIAC 000218 VA-CIRT port 34555
64	62.10.172.146	MY.NET.97.101	WinGate 8080 Attempt
64	MY.NET.97.176	MY.NET.101.192	SNMP public access
61	63.14.215.174	MY.NET.253.105	WinGate 8080 Attempt
60	159.226.45.3	MY.NET.253.42	Watchlist 000222 NET-NCFC
59	172.166.188.140	MY.NET.253.105	WinGate 8080 Attempt
53	MY.NET.98.118	MY.NET.101.192	SNMP public access
52	205.188.179.33	MY.NET.105.2	Attempted Sun RPC high port access
51	172.142.83.53	MY.NET.253.105	WinGate 8080 Attempt
50	159.226.5.222	MY.NET.100.230	Watchlist 000222 NET-NCFC
50	159.226.63.200	MY.NET.100.230	Watchlist 000222 NET-NCFC
50	159.226.21.3	MY.NET.100.230	Watchlist 000222 NET-NCFC
50	172.155.237.188	MY.NET.253.105	WinGate 8080 Attempt
49	209.49.30.23	MY.NET.253.105	WinGate 8080 Attempt
49	159.226.5.65	MY.NET.100.230	Watchlist 000222 NET-NCFC
48	MY.NET.98.148	MY.NET.101.192	SNMP public access
46	63.14.215.127	MY.NET.253.105	WinGate 8080 Attempt
46	205.217.233.122	MY.NET.99.85	WinGate 8080 Attempt
45	216.164.225.93	MY.NET.253.105	WinGate 8080 Attempt
44	208.184.216.191	MY.NET.201.2	Napster 8888 Data
40	158.103.132.100	MY.NET.253.105	WinGate 8080 Attempt
38	MY.NET.97.214	MY.NET.101.192	SNMP public access
37	159.226.66.130	MY.NET.253.42	Watchlist 000222 NET-NCFC
37	203.155.129.248	MY.NET.97.27	WinGate 1080 Attempt
36	159.226.228.1	MY.NET.100.230	Watchlist 000222 NET-NCFC
36	209.10.218.251	MY.NET.60.11	WinGate 1080 Attempt
36	64.38.32.54	MY.NET.253.105	WinGate 8080 Attempt
35	159.226.159.1	MY.NET.253.43	Watchlist 000222 NET-NCFC
34	24.40.25.20	MY.NET.253.105	WinGate 8080 Attempt
32	216.164.230.12	MY.NET.253.105	WinGate 8080 Attempt

```
   32      159.226.66.130       MY.NET.253.43     Watchlist 000222 NET-NCFC
   30      216.179.0.37         MY.NET.60.16      WinGate 1080 Attempt
   30      159.226.66.130       MY.NET.253.41     Watchlist 000222 NET-NCFC
   30      64.38.32.127         MY.NET.253.105    WinGate 8080 Attempt
   30      168.120.16.250       MY.NET.97.143     WinGate 1080 Attempt
(truncated)
```

This table shows which IP address sources are interested in your network and how often they targeted specific hosts with a given method. The top 15 sources includes Israel, China, AmericaOnline, Korea, @Home, National Institute of Health. Most of them are probably sources that you do not want involved with your network at all. Notice also that there are occasions where the source of the attacks are the MY.NET network. Does this mean that some of the MY.NET hosts are compromised? Is it mostly just bad security practice? Or is it case of false positives? To better answer these questions, Table 4 presents the types of alerts generated by hosts inside your network and the frequency of occurrence. The detailed analysis section will address each of these types.

Table 3: Summary of alerts with MY.NET as the source

Alert type	# of occurrences
SNMP public access	1188
SMB Name Wildcard	231
PING-ICMP Destination Unreachable	673
Napster	81
TELNET - Login Incorrect	7
FTP-bad-login	1
TELNET - daemon-active	1
WinGate 1080 Attempt	1

I am not including another table that I produced which contains the number of attacks from a specific host to a specific destination, regardless of the attack method. The reason is that the top part of the table is very similar to Table 2, which means that targeted hosts were not attacked with a series of methods, at least for the item with large number of attacks.

The sorted list of most often attacked hosts on your network are presented in Table 4.

Table 4: The percentage and number of attacks to one certain host

```
=================================================================
      #  of
  %   attacks     to                      type
=================================================================
6.46   5202   MY.NET.217.114      Watchlist 000220 IL-ISDNNET-990517
5.37   4323   MY.NET.217.38       Watchlist 000220 IL-ISDNNET-990517
3.62   2912   MY.NET.253.105      WinGate 8080 Attempt
2.78   2240   MY.NET.217.126      Attempted Sun RPC high port access
2.65   2133   MY.NET.253.41       Watchlist 000222 NET-NCFC
2.25   1808   MY.NET.182.94       Watchlist 000220 IL-ISDNNET-990517
2.11   1702   MY.NET.179.51       Watchlist 000220 IL-ISDNNET-990517
1.48   1188   MY.NET.101.192      SNMP public access
1.45   1170   MY.NET.98.179       IDS247 - MISC - Large UDP Packet
1.04   838    MY.NET.100.230      Watchlist 000222 NET-NCFC
1.01   811    MY.NET.253.43       Watchlist 000222 NET-NCFC
0.93   745    MY.NET.53.28        Watchlist 000220 IL-ISDNNET-990517
0.87   703    MY.NET.253.42       Watchlist 000222 NET-NCFC
0.38   305    MY.NET.60.11        WinGate 1080 Attempt
```

```
0.33    262     MY.NET.60.8          WinGate 1080 Attempt
0.31    249     MY.NET.201.2         Napster 8888 Data
0.28    228     MY.NET.101.192       SMB Name Wildcard
0.20    160     MY.NET.60.16         WinGate 1080 Attempt
0.17    136     MY.NET.97.101        WinGate 8080 Attempt
0.17    133     MY.NET.6.7           Watchlist 000222 NET-NCFC
0.15    118     MY.NET.253.24        GIAC 000218 VA-CIRT port 34555
0.11    90      MY.NET.97.204        Napster 7777 Data
0.11    85      MY.NET.181.88        Watchlist 000220 IL-ISDNNET-990517
0.10    77      MY.NET.99.85         WinGate 8080 Attempt
0.09    76      MY.NET.253.24        GIAC 000218 VA-CIRT port 35555
0.09    75      MY.NET.100.165       Watchlist 000222 NET-NCFC
0.09    73      MY.NET.20.10         WinGate 8080 Attempt
0.08    64      MY.NET.151.33        Watchlist 000220 IL-ISDNNET-990517
0.08    61      MY.NET.97.27         WinGate 1080 Attempt
0.07    56      MY.NET.99.51         WinGate 1080 Attempt
0.07    54      MY.NET.105.2         Attempted Sun RPC high port access
```
(truncated)

From the knowledge of your network (IP Vs Name), and from Table 4, you should be able to quickly identify if some of your critical assets are top targets.

Overall statistics of Snort scan reports

This section presents overall statistics produced by processing the complete set of Snort scan reports. It is meant to give a top view of suspicious scanning activity on your network. Specific events will be discussed in the detailed section.
Scanning destination host ports is a method to gather information on the destination network. Scanning sources are looking for alive hosts, legitimate services running on hosts, and of course the presence of trojans.

All the of port scans detected by the Snort IDS installed on your network were triggered by two family of rules. The first one is that a number of hits, over a period of time, was beyond a specified threshold, e.g. 7 connections in 2 seconds. The other is that the packets had characteristics often used by malicious scanners to penetrate network perimeter defense, for instance, setting both the SYN and the FIN flags in a TCP packets. This is not normal TCP behavior, and some packet filtering devices overlook this type of traffic. Snort alerts uses the word stealth to describe this type of scan.

A total of 1487 port scan events was reported by Snort.

Table 5 presents the main sources scanning hosts and ports on your network. Additional information about the source IP is shown in the last column.

Table 5: Statistics on main port scanning sources.

Source IP	Target_hosts	# of TCP ports	# of UDP ports	Information on source
212.170.19.199	29326	30227	2	Telefonica Data Espana
211.60.222.33	23511	23591	0	NS.MKIBI.COM (Korea?)
169.229.88.203	23503	23638	1	ida-88-203.Reshall.Berkeley.EDU
24.2.123.9	22924	0	22976	ci196729-a.wllmsn1.tn.home.com

209.61.158.214	22035	22114	0	Rackspace.com
202.0.178.98	19627	19793	29	gatnoxs.com
211.36.253.174	17785	17832	0	Korea
211.38.95.138	16307	16366	0	Korea
24.31.224.110	15729	15867	1	mkc-31-224-110.kc.rr.com
193.251.15.20	12951	13603	0	ATuileries-101-2-2-20.abo.wanadoo.fr
24.7.157.43	12625	12822	0	cx149768-a.stche1.la.home.com
211.112.142.2	12452	12512	0	Korea
128.125.104.191	9636	9670	0	avila.usc.edu
4.54.218.36	9625	9927	0	BBN Planet
63.29.27.192	8344	8574	0	UUNET Technologies, Inc
35.10.82.105	6367	6404	0	mcc-2.user.msu.edu
4.54.218.182	5219	5360	0	PPPa57-ResaleAppleton2-2R7237.saturn.bbn.com
200.241.187.2	5106	5136	12	Brazil
207.155.88.200	4636	4783	3	
63.79.70.130	3049	3056	0	falcon.jdenisco.com
202.147.24.142	2580	2634	0	me-as-07-142.free.net.au
4.54.218.59	2185	2306	0	PPPa30-ResaleAppleton1-1R7237.saturn.bbn.com
211.44.13.212	1561	1562	0	NS2.AGENTLEADER.COM
195.132.120.31	1055	1127	0	r120m31.cybercable.tm.fr

Again, a quick look at the sources should raise concerns. You have many suspicious sources scanning the ports of your hosts, the next question should be: what are they looking for? To help answer this question I built Table 6. It lists the most popular destination ports and the probable reasons why they are targeted.

Table 6: The most popular ports scanned.

Port number TCP/UDP	# of hits	Looking for?
21	141623	File Transfer Protocol (FTP)
53	58162	Domain Name Service (DNS)
27374	36818	Subseven trojan
98	34631	Linuxconf (sysadmin tool for Linux)
110	1572	POP3 (Internet mail)
6970	1374	RealAudio (false positive)
23	530	Telnet (remote login)
31337	473	Back Orifice trojan
44767	430	
12345	225	Netbus trojan
7000	187	
6346	163	
1693	155	
1685	153	
1686	151	
1682	144	
1678	138	

1681	137	
2929	128	
1059	126	

This is a fairly typical lists of scanned ports. The targeted ports represent what we see elsewhere a lot, i.e. trying to access either well know ports (0-1023) often offering services outside like FTP, Telnet and DNS, as well as fishing for trojans servers on their default listening ports. There are also some false positive alerts in this list. It is the case for the port used by RealAudio (UDP 6970) and for some legitimate DNS traffic.

The next question we should try to answer is which hosts are targeted by port scans. Table 7 presents the top 20 destination host subjected to port scans and the number of port hits. To put the number of hit of Table 7 in perspective, it is interesting to note that 99.7% of your hosts that were port scanned had ports hit counts below 30.

Table 7: Top 20 hosts targeted by port scans

Host	Hits
MY.NET.70.127	1389
MY.NET.97.83	1242
MY.NET.253.114	1147
MY.NET.98.118	1060
MY.NET.181.88	903
MY.NET.1.3	899
MY.NET.101.89	843
MY.NET.5.4	526
MY.NET.98.185	360
MY.NET.60.8	358
MY.NET.179.56	336
MY.NET.179.86	332
MY.NET.60.11	331
MY.NET.97.68	328
MY.NET.179.78	312
MY.NET.153.111	270
MY.NET.153.112	242
MY.NET.156.109	210
MY.NET.156.104	209
MY.NET.156.106	203

Some of these port scans will be discussed in the detailed analysis section.

Snort also reported scans coming from inside your network. Table 8 lists all these suspicious events. These could be a signs of compromised hosts. More on these in the detailed analysis section.

Table 8: Port scans alerts from inside hosts.

Source	# of port scan alerts
MY.NET.1.3	75
MY.NET.101.192	2
MY.NET.1.4	1
MY.NET.19.10	1
MY.NET.101.1	1

Detailed analysis of specific alerts and specific hosts activities

This section presents detailed analysis of specific security events highlighted in the overall statistics sections. The four sub-sections cover alerts and port scans, from inside and outside hosts. My conclusion is shown at the end of each discussion, within brackets, e.g. **[conclusion].**

Detailed Analysis - Alerts from outside hosts

This section presents some analysis of Snort alerts generated by hosts outside your network. Obviously, most of the alerts come from outside hosts. I will discuss the most often used attacks in this section. Statistics on the frequency of these attacks were shown earlier in Table 1.

· **20067 SYN-FYN scans (see Table 1)**

These are reconnaissance scans using a stealthy techniques to evade being detected by perimeter control devices (firewalls, routers, etc…)

· **Watchlist 000220 IL-ISDNNET-990517 (see Table 1)**

This is traffic involving hosts in Israel, from which suspicious activity has been seen before by the network security community. This is all inbound traffic alerts. Most of the traffic involve ports 6699 and 6698, which looks like Napster traffic, but I am not certain. You would have to confirm if this is desired traffic with further investigation.

· **Watchlist 000222 NET-NCFC (see Table 1)**

This is traffic involving hosts at the Chinese Academy of Science. The alerts are all from inbound traffic. Most of this traffic is to destination port 25, which is SMTP. There is probably email exchange with these Chinese hosts. You should clarify this email activity on your affected hosts. There is also some Telnet traffic in this watch list.

Sample SMTP activity:
08/04-04:48:40.674427 [**] Watchlist 000222 NET-NCFC [**] 159.226.91.37:3042 -> MY.NET.100.230:25
08/04-04:48:42.631850 [**] Watchlist 000222 NET-NCFC [**] 159.226.91.37:3040 -> MY.NET.100.230:25
08/04-04:48:43.532486 [**] Watchlist 000222 NET-NCFC [**] 159.226.91.37:3040 -> MY.NET.100.230:25

08/04-04:48:44.373603 [**] Watchlist 000222 NET-NCFC [**] 159.226.91.37:3040 -> MY.NET.100.230:25

· **WinGate 8080 and 1080 Attempt (see Table 1)**

These are attempts to find WinGate proxy servers on your network. Once found, the attacker would try to compromise your host with a WinGate vulnerability and, if successful, could use your host to attack other hosts. I'm not sure if those attempts were successful or not. For instance, the patterns show the same outside host generating many WinGate alerts on the same destination host like this:

06/27-13:25:45.386994 [**] WinGate 8080 Attempt [**] 24.6.132.179:1308 -> MY.NET.253.105:8080
06/27-13:25:52.639126 [**] WinGate 8080 Attempt [**] 24.6.132.179:1310 -> MY.NET.253.105:8080
06/27-13:26:09.669794 [**] WinGate 8080 Attempt [**] 24.6.132.179:1313 -> MY.NET.253.105:8080
06/27-13:26:09.669924 [**] WinGate 8080 Attempt [**] 24.6.132.179:1314 -> MY.NET.253.105:8080
06/27-13:26:09.689942 [**] WinGate 8080 Attempt [**] 24.6.132.179:1315 -> MY.NET.253.105:8080
06/27-13:26:09.699518 [**] WinGate 8080 Attempt [**] 24.6.132.179:1316 -> MY.NET.253.105:8080
06/27-13:26:09.707177 [**] WinGate 8080 Attempt [**] 24.6.132.179:1317 -> MY.NET.253.105:8080
06/27-13:26:09.707270 [**] WinGate 8080 Attempt [**] 24.6.132.179:1318 -> MY.NET.253.105:8080
06/27-13:26:09.712752 [**] WinGate 8080 Attempt [**] 24.6.132.179:1319 -> MY.NET.253.105:8080
…

This does not look like a scan, but more like some traffic involving a server. You should identify the services on your host MY.NET.253.105, as it is almost always the destination.

· **Attempted Sun RPC high port access (see Table 1)**

08/04-10:00:31.105746 [**] Attempted Sun RPC high port access [**] 205.188.153.111:4000 -> MY.NET.217.126:32771
08/04-10:01:31.000604 [**] Attempted Sun RPC high port access [**] 205.188.153.111:4000 -> MY.NET.217.126:32771
08/04-10:02:30.954425 [**] Attempted Sun RPC high port access [**] 205.188.153.111:4000 -> MY.NET.217.126:32771
08/04-10:03:30.782653 [**] Attempted Sun RPC high port access [**] 205.188.153.111:4000 -> MY.NET.217.126:32771
08/04-10:05:30.563818 [**] Attempted Sun RPC high port access [**] 205.188.153.111:4000 -> MY.NET.217.126:32771
08/04-10:06:30.487499 [**] Attempted Sun RPC high port access [**] 205.188.153.111:4000 -> MY.NET.217.126:32771

These alerts are caused by accesses to a RPC high port (32771). The recurring pattern almost always involve the source port 4000. This is probably ICQ traffic (port 4000). Is this a security concern? You have to decide if you have a real requirement for ICQ. If not, then I would suggest you block ICQ traffic because ICQ clients are exploitable because of many published vulnerabilities.

· **SNMP public access (see Table 1)**

07/11-16:00:18.945847 [**] SNMP public access [**] MY.NET.97.122:1125 -> MY.NET.101.192:161
07/11-16:00:23.891592 [**] SNMP public access [**] MY.NET.97.122:1125 -> MY.NET.101.192:161
07/11-16:01:26.018511 [**] SNMP public access [**] MY.NET.97.122:1128 -> MY.NET.101.192:161
07/10-18:14:05.658622 [**] SNMP public access [**] MY.NET.97.159:1042 -> MY.NET.101.192:161
07/10-18:14:07.349004 [**] SNMP public access [**] MY.NET.97.159:1043 -> MY.NET.101.192:161

07/10-18:14:07.557554 [**] SNMP public access [**] MY.NET.97.159:1043 -> MY.NET.101.192:161
07/10-18:16:11.703651 [**] SNMP public access [**] MY.NET.97.159:1057 -> MY.NET.101.192:161
07/10-18:17:13.553745 [**] SNMP public access [**] MY.NET.97.159:1059 -> MY.NET.101.192:161
…

All these alerts involve MY.NET.101.192 on the normal SNMP port (Simple Network Management Protocol – port 161). The other hosts are always from inside your network, which is reassuring. The alert is because the transactions with this SNMP device use the SNMP community string of "public", which like using a default password. This password should be change to increase security because some SNMP devices are critical infrastructure assets, like routers.

· **IDS247 - MISC - Large UDP Packet (see Table 1)**

08/05-18:57:26.372376 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 -> MY.NET.98.179:6970
08/05-18:57:28.174478 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 -> MY.NET.98.179:6970
08/05-18:57:29.402770 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 -> MY.NET.98.179:6970
08/05-18:57:30.987902 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 -> MY.NET.98.179:6970
08/05-18:57:32.242808 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 -> MY.NET.98.179:6970
08/05-18:57:34.750281 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 ->
…

Port 6970 indicates probable RealAudio traffic. Not a security concern.

· **Napster 8888 Data and Napster 7777 Data (see Table 1)**

Napster traffic. Do you want to allow Napster traffic on your network. This Napster issue is discussed at a few other places in this report.

· **SMB Name Wildcard (see Table 1)**

These are all the SMB Name Wildcard alerts with outside sources:
08/03-11:38:50.220540 [**] SMB Name Wildcard [**] 206.15.45.126:137 -> MY.NET.70.66:137
08/04-16:23:34.611008 [**] SMB Name Wildcard [**] 132.201.232.167:137 -> MY.NET.15.127:137
08/04-16:23:34.611091 [**] SMB Name Wildcard [**] 208.16.237.10:137 -> MY.NET.15.127:137
08/04-16:23:42.770412 [**] SMB Name Wildcard [**] 208.16.237.10:137 -> MY.NET.15.127:137
08/04-16:23:42.770527 [**] SMB Name Wildcard [**] 132.201.232.167:137 -> MY.NET.15.127:137
08/04-16:23:44.293564 [**] SMB Name Wildcard [**] 132.201.232.167:137 -> MY.NET.15.127:137
08/04-16:23:44.293784 [**] SMB Name Wildcard [**] 208.16.237.10:137 -> MY.NET.15.127:137
08/04-16:23:50.888340 [**] SMB Name Wildcard [**] 132.201.232.167:137 -> MY.NET.15.127:137
08/04-16:23:50.888445 [**] SMB Name Wildcard [**] 208.16.237.10:137 -> MY.NET.15.127:137

This is not the kind of traffic that you want to see, i.e. NetBIOS traffic (port 137 to port 137) involving hosts outside your network. This is Windows networking traffic and it should normally be contained within your intranet. I researched information on the three external IP sources. The results are not known malicious sources. Two of them are large ISPs (Sprint and Southwestern Bell) and the third is a public

library in xxxxx county (sanitized). Could this be inside users generating hits from their homes? Is this public library in your geographical area?

As a good practice, I recommend that you block inbound traffic going to Windows networking ports (137-139).

· **GIAC 000218 VA-CIRT port 34555 and GIAC 000218 VA-CIRT port 35555 (see Table 1)**

This alert refers to malicious activity reported on the February 18th 2000, at the Global Incident Analysis Center (GIAC) of the SANS Institute (http://www.sans.org/giac.htm). The GIAC report describes the communications involving a trojan (Windows version of Trinoo) used for distributed denial of service attack. The complete signature is a ping to an inside host on port 34555 provoking a reply from port 35555.

Sample alerts:
06/30-14:55:27.290209 [**] GIAC 000218 VA-CIRT port 34555 [**] 140.142.100.15:25 -> MY.NET.253.52:34555
06/30-14:55:27.476759 [**] GIAC 000218 VA-CIRT port 34555 [**] 140.142.100.15:25 -> MY.NET.253.52:34555
06/30-14:55:27.724280 [**] GIAC 000218 VA-CIRT port 34555 [**] 140.142.100.15:25 -> MY.NET.253.52:34555
06/30-17:31:33.953871 [**] GIAC 000218 VA-CIRT port 35555 [**] 199.1.13.9:113 -> MY.NET.6.35:35555
06/30-17:31:33.953930 [**] GIAC 000218 VA-CIRT port 35555 [**] 199.1.13.9:113 -> MY.NET.6.35:35555
06/30-17:31:34.018246 [**] GIAC 000218 VA-CIRT port 35555 [**] 199.1.13.9:113 -> MY.NET.6.35:35555
06/30-17:31:34.018457 [**] GIAC 000218 VA-CIRT port 35555 [**] 199.1.13.9:113 -> MY.NET.6.35:35555
…

These alerts were all generated simply because the port 34555 and 35555 were active. Since none of the alerts have an inside host sending from any of these two ports I considered this to be false positive.

· **Possible wu-ftpd exploit - GIAC000623 (see Table 1)**
· **site exec - Possible wu-ftpd exploit - GIAC000623 (see Table 1)**

06/30-16:33:57.773279 [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**] 151.164.223.206:4499 -> MY.NET.99.16:21
06/30-16:34:00.037398 [**] Possible wu-ftpd exploit - GIAC000623 [**] 151.164.223.206:4499 -> MY.NET.99.16:21
06/30-16:35:11.406398 [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**] 151.164.223.206:4500 -> MY.NET.144.59:21
06/30-16:35:13.560305 [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**] 151.164.223.206:4500 -> MY.NET.144.59:21
06/30-16:35:13.626498 [**] Possible wu-ftpd exploit - GIAC000623 [**] 151.164.223.206:4500 -> MY.NET.144.59:21
07/19-03:53:00.191779 [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**] 212.35.163.64:1245 -> MY.NET.100.165:21
07/29-12:07:56.525800 [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**] 211.38.95.138:3048 -> MY.NET.156.127:21

These alerts concern me. GIAC reported a new FTP attack (wu-ftpd exploit) only seven days before seeing these alerts on your network. The Snort rule is fairly specific in this case, it includes targeting the FTP control port (21) with a specific string in the content. There is less chance of a false positive in this situation. Let's investigate the each of your three inside hosts affected..

Investigation on MY.NET.156.127:

Attacking source 211.38.95.138 is from Korea. It scanned your network only once. It was complete network (MY.NET.x.y) scan with destination port 21. It scanned MY.NET.156.27 on Jul 29th at 12:00:57. The "site exec - Possible wu-ftpd exploit" alert for this host is reported about 7 minutes later.

Jul 29 12:02:49 211.38.95.138:2971 -> MY.NET.156.127:21 SYN **S*****

There is a good probability that MY.NET.156.27 is compromised. Although Snort did not report anything involving MY.NET.156.27 after the possible compromising event, you must investigate this host immediately.

Investigation on MY.NET.100.165:

Here is a scary trace:

07/19-03:53:00.191779 [**] site exec - Possible wu-ftpd exploit - GIAC000623 [**] 212.35.163.64:1245 -> MY.NET.100.165:21
07/30-22:49:42.452934 [**] Watchlist 000222 NET-NCFC [**] 159.226.5.152:719 -> MY.NET.100.165:80
07/30-22:49:43.244400 [**] Watchlist 000222 NET-NCFC [**] 159.226.5.152:719 -> MY.NET.100.165:80
07/30-22:49:44.554166 [**] Watchlist 000222 NET-NCFC [**] 159.226.5.152:719 -> MY.NET.100.165:80
07/30-22:49:44.560012 [**] Watchlist 000222 NET-NCFC [**] 159.226.5.152:719 -> MY.NET.100.165:80
…
07/30-22:51:58.267475 [**] Watchlist 000222 NET-NCFC [**] 159.226.5.152:719 -> MY.NET.100.165:80
07/30-22:51:58.267525 [**] Watchlist 000222 NET-NCFC [**] 159.226.5.152:719 -> MY.NET.100.165:80
07/30-22:51:58.801371 [**] Watchlist 000222 NET-NCFC [**] 159.226.5.152:719 -> MY.NET.100.165:80
07/30-22:51:59.131277 [**] Watchlist 000222 NET-NCFC [**] 159.226.5.152:719 -> MY.NET.100.165:80
07/30-22:51:59.466739 [**] Watchlist 000222 NET-NCFC [**] 159.226.5.152:719 -> MY.NET.100.165:80

On July 19th MY.NET.100.165 may have been compromised with a wu-ftpd exploit by 212.35.163.64:1245 (Ukraine). On July 30th , a series of alerts related to traffic between MY.NET.100.165 and a host of the Chinese Academy of Sciences. This traffic is very suspicious because it is from low port 719) to low port (80). It looks almost like a server was installed on MY.NET.100.165, a web server? **MY.NET.100.165 is most likely compromised. It should be shut down and investigated.**

Investigation on MY.NET. 99.16 and MY.NET. 144.59:

I did not find any relevant alerts, beyond the possible wu-ftpd exploit alerts, that would involve these two local hosts. Considering the discussions on MY.NET.156.27 and MY.NET.100.165 and possible wu-ftpd exploits above, it would be wise to investigate MY.NET. 99.16 and MY.NET. 144.59 as well.

Detailed Analysis - Alerts from inside hosts

· **SMB Name Wildcard (see Table 1)**

07/27-20:12:29.816023 [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137
07/27-20:12:31.306913 [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137
07/27-20:13:48.439202 [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137
07/27-20:13:49.969821 [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137
07/30-18:44:40.664758 [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137
07/30-18:44:42.151370 [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137
07/30-18:44:43.750890 [**] SMB Name Wildcard [**] MY.NET.101.160:137 -> MY.NET.101.192:137

This kind of internal traffic is most likely normal NetBIOS traffic. It almost always involve MY.NET.101.192 on port 137 (NetBIOS name service)

· **One WinGate 1080 Attempt by MY.NET.99.51:20 (see Table 3)**

06/27-13:20:12.138930 [**] WinGate 1080 Attempt [**] MY.NET.99.51:20 -> MY.NET.101.155:1080

Could this be normal internal FTP traffic (port 20 = data) between two internal hosts, MY.NET.99.51 being the server.

· **One TELNET - daemon-active - MY.NET.99.51:20 (see Table 3)**

08/05-19:03:45.522918 [**] IDS08 - TELNET - daemon-active [**] MY.NET.99.51:23 -> 24.25.111.117:1029

I do not know for sure what this alert could be. Could MY.NET.99.51 be also a Telnet server. Do you want to allow Telnet from outside? This one would require further analysis.

· **Various Napster alerts from MY.NET.x.y (see Table 3)**

You have hosts with Napster on your net. The most Napster busy clients for that period are MY.NET.201.2 and MY.NET.217.158

· **TELNET - Login Incorrect from MY.NET.x.y (see Table 3)**

08/05-18:37:37.745999 [**] IDS127 - TELNET - Login Incorrect [**] MY.NET.60.11:23 -> 208.198.33.168:1024
08/05-18:47:13.982488 [**] IDS127 - TELNET - Login Incorrect [**] MY.NET.60.8:23 -> 207.172.151.22:1674
08/05-18:47:20.888622 [**] IDS127 - TELNET - Login Incorrect [**] MY.NET.60.8:23 -> 207.172.151.22:1674
08/05-18:53:09.934812 [**] IDS127 - TELNET - Login Incorrect [**] MY.NET.60.8:23 -> 151.198.144.196:1026
08/05-18:54:24.387218 [**] IDS127 - TELNET - Login Incorrect [**] MY.NET.60.11:23 -> 24.6.134.169:3452
08/05-18:56:11.376893 [**] IDS127 - TELNET - Login Incorrect [**] MY.NET.60.8:23 -> 63.24.126.127:1197
08/05-18:59:23.821523 [**] IDS127 - TELNET - Login Incorrect [**] MY.NET.6.7:23 -> 38.30.171.95:1223

These are failed attempts to telnet from outside hosts to inside hosts. It could be honest mistakes, like typing errors. However, you should consider blocking Telnet from outside if you do not have a requirement for it.

Detailed Analysis - Port scans from outside hosts

· **MY.NET.70.127 was hit 1389 times by outside hosts doing port scans (see Table 7)**
Scanned on large sequences of high ports.
[Reconnaissance scan]

· **MY.NET.97.83 was hit 1242 times by outside hosts doing port scans (see Table 7)**

Jul 8 15:25:32 165.138.228.4:7777 -> MY.NET.97.83:1700 UDP
Jul 8 15:25:35 165.138.228.4:7777 -> MY.NET.97.83:1700 UDP
Jul 8 15:25:35 165.138.228.4:7777 -> MY.NET.97.83:1681 UDP
Jul 8 15:25:35 165.138.228.4:7777 -> MY.NET.97.83:1682 UDP
Jul 8 15:25:35 165.138.228.4:7777 -> MY.NET.97.83:1685 UDP
Jul 8 15:25:35 165.138.228.4:7777 -> MY.NET.97.83:1686 UDP
Jul 8 15:25:35 165.138.228.4:7777 -> MY.NET.97.83:1678 UDP
Jul 8 15:25:35 165.138.228.4:7777 -> MY.NET.97.83:1693 UDP

This is not a port scan but Napster traffic. Napster is a program that allows two hosts to exchange data directly between each other. The two Napster hosts first make contact through a central Napster server. Napster has been in the news lately because it is used extensively to illegally exchange copyrighted material (music files in MP3 format). One of the side effect of having many Napster clients on a network is large bandwidth consumption. Although it is a false positive from the port scan point of view, it points to a type of traffic, Napster, that you may want deny on your internal network.
[False positive port scan – Napster traffic]

· **MY.NET.253.114 was hit 1147 times by outside hosts doing port scans (see Table 7)**

Large port scan.
[Reconnaissance scan]

· **MY.NET. 98.118 was hit 1060 times by outside hosts doing port scans (see Table 7)**

Mostly scanning of low port numbers
[Reconnaissance scan]

· **MY.NET.181.88 was hit 903 times by outside hosts doing port scans (see Table 7)**
· **MY.NET.179.56 was hit 336 times by outside hosts doing port scans (see Table 7)**
· **MY.NET.179.86 was hit 332 times by outside hosts doing port scans (see Table 7)**
· **MY.NET.179.78 was hit 312 times by outside hosts doing port scans (see Table 7)**

Typical reconnaissance port scans.
[Reconnaissance scan]

· **MY.NET.1.3 was hit 899 times by outside hosts doing port scans (see Table 7)**

See discussion titled *75 port scan alerts with source = MY.NET.1.3* in the section Detailed Analysis - Port scans from inside hosts.
[False positive – DNS server]

· **MY.NET. 101.89 was hit 843 times by outside hosts doing port scans (see Table 7)**

See discussion titled *75 port scan alerts with source = MY.NET.1.3* in the section Detailed Analysis - Port scans from inside hosts.
[False positive – DNS client]

· **MY.NET.5.4 was hit 526 times by outside hosts doing port scans (see Table 7)**

Scanning of mix of low and high port numbers.
[Reconnaissance scan]

· **MY.NET.98.185 was hit 360 times by outside hosts doing port scans (see Table 7)**

Scans of high UDP ports.
[Reconnaissance scan]

· **MY.NET.60.8 was hit 358 times by outside hosts doing port scans (see Table 7)**

sample:
Jul 27 10:14:06 210.84.179.196:1248 -> MY.NET.60.8:194 SYN **S*****
Jul 27 10:14:06 210.84.179.196:1249 -> MY.NET.60.8:195 SYN **S*****
Jul 27 10:14:06 210.84.179.196:1250 -> MY.NET.60.8:196 SYN **S*****
Jul 27 10:14:06 210.84.179.196:1251 -> MY.NET.60.8:197 SYN **S*****
Jul 27 10:14:06 210.84.179.196:1252 -> MY.NET.60.8:198 SYN **S*****
Jul 27 10:14:06 210.84.179.196:1254 -> MY.NET.60.8:200 SYN **S*****
Jul 27 10:14:06 210.84.179.196:1253 -> MY.NET.60.8:199 SYN **S*****
Jul 27 10:15:14 210.84.179.196:15392 -> MY.NET.60.8:113 SYN **S*****
Jul 27 10:15:14 210.84.179.196:15396 -> MY.NET.60.8:113 SYNFIN **SF****
Jul 27 10:15:14 210.84.179.196:15398 -> MY.NET.60.8:113 SYN 21S***** RESERVEDBITS
Jul 27 10:15:14 210.84.179.196:15394 -> MY.NET.60.8:113 FIN ***F****
Jul 27 10:15:14 210.84.179.196:15397 -> MY.NET.60.8:113 VECNA *****P**
…

Scans of high port numbers. There is an interesting pattern on destination port 113. Five different TCP packet formats are tried. This is probably the signature of a scanning tool.
[Reconnaissance scan]

· **MY.NET.60.11 was hit 331 times by outside hosts doing port scans (see Table 7)**

Mostly scans of high port numbers.
[Reconnaissance scan]

· **MY.NET.97.68 was hit 328 times by outside hosts doing port scans (see Table 7)**

Jul 9 21:26:09 165.138.228.4:7777 -> MY.NET.97.68:2077 UDP
Jul 9 21:26:09 165.138.228.4:7777 -> MY.NET.97.68:2079 UDP
Jul 9 21:26:09 165.138.228.4:7777 -> MY.NET.97.68:2081 UDP
Jul 9 21:26:09 165.138.228.4:7777 -> MY.NET.97.68:2084 UDP
Jul 9 21:26:09 165.138.228.4:7777 -> MY.NET.97.68:2089 UDP
Jul 9 21:26:09 165.138.228.4:7777 -> MY.NET.97.68:2091 UDP
Jul 9 21:26:09 165.138.228.4:7777 -> MY.NET.97.68:2092 UDP
Jul 9 21:26:09 165.138.228.4:7777 -> MY.NET.97.68:2095 UDP
…

Mostly Napster traffic. Also scans of UDP ports

[Napster traffic + Reconnaissance scan]

- **MY.NET.153.109 was hit 133 times by outside hosts doing port scans**
- **MY.NET.153.111 was hit 270 times by outside hosts doing port scans (see Table 7)**
- **MY.NET.153.112 was hit 242 times by outside hosts doing port scans (see Table 7)**

Napster traffic for the 3 alerts above.
[False positive port scan – Napster traffic]

- **MY.NET.156.101 was hit 194 times by outside hosts doing port scans**
- **MY.NET.156.103 was hit 158 times by outside hosts doing port scans**
- **MY.NET.156.104 was hit 209 times by outside hosts doing port scans (see Table 7)**
- **MY.NET.156.105 was hit 194 times by outside hosts doing port scans**
- **MY.NET.156.106 was hit 203 times by outside hosts doing port scans (see Table 7)**
- **MY.NET.156.109 was hit 210 times by outside hosts doing port scans (see Table 7)**
- **MY.NET.156.110 was hit 127 times by outside hosts doing port scans**
- **MY.NET.156.111 was hit 203 times by outside hosts doing port scans**

The following trace applies to the eight port scans above:

```
Jun 27 13:48:21 216.46.175.35:29160 -> MY.NET.156.111:6970 UDP
Jun 27 13:48:23 216.46.175.35:22594 -> MY.NET.156.101:6970 UDP
Jun 27 13:48:23 216.46.175.35:24328 -> MY.NET.156.106:6970 UDP
Jun 27 13:48:23 216.46.175.35:15504 -> MY.NET.156.109:6970 UDP
Jun 27 13:48:23 216.46.175.35:8708 -> MY.NET.156.107:6970 UDP
Jun 27 13:48:23 216.46.175.35:24684 -> MY.NET.156.105:6970 UDP
Jun 27 13:48:23 216.46.175.35:23638 -> MY.NET.156.104:6970 UDP
Jun 27 13:48:23 216.46.175.35:22504 -> MY.NET.156.103:6970 UDP
Jun 27 13:48:26 216.46.175.35:24684 -> MY.NET.156.105:6970 UDP
Jun 27 13:48:26 216.46.175.35:8708 -> MY.NET.156.107:6970 UDP
Jun 27 13:48:26 216.46.175.35:15504 -> MY.NET.156.109:6970 UDP
Jun 27 13:48:26 216.46.175.35:29160 -> MY.NET.156.111:6970 UDP
Jun 27 13:48:26 216.46.175.35:24328 -> MY.NET.156.106:6970 UDP
…
```

This looks like normal RealAudio traffic (UDP 6970). RealAudio client allows the user to receive streaming audio/video. The complete trace indicates that these eight hosts, with close IP numbers, were receiving RealAudio traffic at the same time. This could be members of the same work unit listening to the same Webcast event.
[False positive port scan – RealAudio traffic]

Detailed Analysis - Port scans from inside hosts

- **75 port scan alerts with source = MY.NET.1.3 (see Table 8)**

The port scan reports show MY.NET.1.3 scanning some hosts inside your network. Is this a compromised hosts? This scans were present the first day of the Snort log, so there is no way to correlate the start of the scans with another alert related to MY.NET.1.3 being infected. The scans behavior need to be investigated in more details.

The characteristics of the packets are always the same: UDP, source port 53, high destination port. This looks like a DNS server. More research on the traffic with its most popular "target" (MY.NET.101.89)

also shows signs of legitimate DNS query traffic. For instance, there is a clear correlation between the port number and the time, which probably means that MY.NET.101.89 is just a busy DNS client.

I conclude that MY.NET.1.3 is not scanning MY.NET.101.89 and the other inside hosts but simply answering DNS clients.
[False positive – DNS server]

· **2 port scan alerts with source = MY.NET.101.192 (see Table 8)**

07/28-18:40:35.376341 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.101.192 (THRESHOLD 7 connections in 2 seconds) [**]
07/30-18:54:23.857420 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.101.192 (THRESHOLD 7 connections in 2 seconds) [**]

Some of the scan events:

Jul 28 18:28:08 MY.NET.101.192:161 -> MY.NET.98.127:1055 UDP
Jul 28 18:28:08 MY.NET.101.192:161 -> MY.NET.98.127:1056 UDP
Jul 28 18:28:08 MY.NET.101.192:161 -> MY.NET.98.127:1057 UDP
Jul 28 18:28:08 MY.NET.101.192:161 -> MY.NET.98.127:1058 UDP
Jul 28 18:28:09 MY.NET.101.192:161 -> MY.NET.98.127:1059 UDP
Jul 28 18:28:10 MY.NET.101.192:161 -> MY.NET.98.127:1060 UDP
Jul 28 18:28:10 MY.NET.101.192:161 -> MY.NET.98.127:1061 UDP
Jul 28 18:28:10 MY.NET.101.192:161 -> MY.NET.98.127:1062 UDP
Jul 28 18:28:12 MY.NET.101.192:161 -> MY.NET.98.127:1063 UDP
Jul 28 18:28:14 MY.NET.101.192:161 -> MY.NET.98.127:1063 UDP
…

There are also many alerts involving MY.NET.101.192 similar to this:

07/14-08:13:26.293120 [**] SNMP public access [**] MY.NET.97.237:1044 -> MY.NET.101.192:161
07/14-08:13:26.355872 [**] SNMP public access [**] MY.NET.97.237:1048 -> MY.NET.101.192:161
07/14-08:13:26.757707 [**] SNMP public access [**] MY.NET.97.237:1049 -> MY.NET.101.192:161
07/14-08:13:27.270174 [**] SNMP public access [**] MY.NET.97.237:1051 -> MY.NET.101.192:161
07/14-08:13:29.240565 [**] SNMP public access [**] MY.NET.97.237:1052 -> MY.NET.101.192:161

The appearance of scanning is probably from legitimate Simple Network Management Protocol (SNMP) traffic. This alert is from a threshold rule, and not from a stealth rule, which would be a bigger sign of a malicious scan. To address the [**] SNMP public access [**] alerts, but the community string should be changed to something more secure.
[Probably false positive, but security measures to be taken - password].

· **1 port scan alert with source = MY.NET.1.4 (see Table 8)**

06/30-10:57:05.273259 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.1.4 (THRESHOLD 7 connections in 2 seconds) [**]

sample scan traces:
Aug 8 11:44:16 MY.NET.1.4:53 -> MY.NET.75.149:1620 UDP
Aug 8 11:44:16 MY.NET.1.4:53 -> MY.NET.75.149:1621 UDP
Aug 8 11:44:16 MY.NET.1.4:53 -> MY.NET.75.106:1847 UDP

Aug 8 11:44:16 MY.NET.1.4:53 -> MY.NET.75.106:1848 UDP
Aug 8 11:44:19 MY.NET.1.4:53 -> MY.NET.75.118:2332 UDP
Aug 8 11:44:19 MY.NET.1.4:53 -> MY.NET.75.118:2333 UDP
…
Jun 30 10:43:55 MY.NET.1.4:53 -> MY.NET.101.53:2393 UDP
Jun 30 10:43:55 MY.NET.1.4:53 -> MY.NET.101.53:2397 UDP
Jun 30 10:43:56 MY.NET.1.4:53 -> MY.NET.101.53:2404 UDP
Jun 30 10:43:56 MY.NET.1.4:53 -> MY.NET.101.53:2406 UDP
Jun 30 10:43:56 MY.NET.1.4:53 -> MY.NET.101.53:2408 UDP

This looks a lot like normal DNS traffic. MY.NET.1.4:53 is probably a DNS server.
[False positive – DNS server]

· **1 port scan alert with source = MY.NET.19.10 (see Table 8)**

06/28-15:51:14.018991 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.19.10 (STEALTH)
06/28-15:51:18.405985 [**] spp_portscan: End of portscan from MY.NET.19.10 (TOTAL HOSTS:1 TCP:1 UDP:0) [**]

The Snort scan report for that day is missing. It is therefore impossible for me to know which host was the target. Note that only one host was hit, and on one port only. The scan rule that triggered this alert is STEALTH, which means that it was a malformed TCP packet. This is very low volume, one hit, but it should be investigated further because it is not normal for this host to send a malformed packet.
[Inconclusive – could be compromised, need to be investigated further]

· **1 port scan alert with source = MY.NET.101.1 (see Table 8)**

06/28-16:20:45.959895 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.101.1 (STEALTH)
06/28-16:20:49.756355 [**] spp_portscan: End of portscan from MY.NET.101.1 (TOTAL HOSTS:1 TCP:1 UDP:0) [**]

The Snort scan report for that day is missing. It is therefore impossible for me to know which host was the target. Note that only one host was hit, and on one port only. The scan rule that triggered this alert is STEALTH, which means that it was a malformed TCP packet. This is very low volume, one hit, but it should be investigated further because it is not normal for this host to send a malformed packet.
[Inconclusive – could be compromised, need to be investigated further]

· **MY.NET.5.37 is scanning its IP neighbors**

Aug 10 06:15:03 MY.NET.5.37:2600 -> MY.NET.5.3:5632 UDP
Aug 10 06:15:03 MY.NET.5.37:2600 -> MY.NET.5.3:22 UDP
Aug 10 06:15:04 MY.NET.5.37:2600 -> MY.NET.5.13:5632 UDP
Aug 10 06:15:04 MY.NET.5.37:2600 -> MY.NET.5.13:22 UDP
Aug 10 06:15:04 MY.NET.5.37:2600 -> MY.NET.5.15:5632 UDP
Aug 10 06:15:04 MY.NET.5.37:2600 -> MY.NET.5.25:5632 UDP
Aug 10 06:15:04 MY.NET.5.37:2600 -> MY.NET.5.29:5632 UDP
Aug 10 06:15:04 MY.NET.5.37:2600 -> MY.NET.5.34:5632 UDP
Aug 10 06:15:05 MY.NET.5.37:2600 -> MY.NET.5.40:5632 UDP
Aug 10 06:15:05 MY.NET.5.37:2600 -> MY.NET.5.41:5632 UDP
Aug 10 06:15:05 MY.NET.5.37:2600 -> MY.NET.5.42:5632 UDP
Aug 10 06:15:05 MY.NET.5.37:2600 -> MY.NET.5.45:5632 UDP
Aug 10 06:15:06 MY.NET.5.37:2600 -> MY.NET.5.101:5632 UDP

Aug 10 06:15:07 MY.NET.5.37:2600 -> MY.NET.5.103:5632 UDP
Aug 10 06:15:07 MY.NET.5.37:2600 -> MY.NET.5.104:5632 UDP
Aug 10 06:15:07 MY.NET.5.37:2600 -> MY.NET.5.105:5632 UDP
Aug 10 06:15:07 MY.NET.5.37:2600 -> MY.NET.5.106:5632 UDP
Aug 10 06:15:07 MY.NET.5.37:2600 -> MY.NET.5.107:5632 UDP
Aug 10 06:15:07 MY.NET.5.37:2600 -> MY.NET.5.109:5632 UDP
Aug 10 06:15:07 MY.NET.5.37:2600 -> MY.NET.5.119:5632 UDP
Aug 10 06:15:07 MY.NET.5.37:2600 -> MY.NET.5.120:5632 UDP
Aug 10 06:15:10 MY.NET.5.37:2600 -> MY.NET.5.200:5632 UDP
Aug 10 06:15:10 MY.NET.5.37:2600 -> MY.NET.5.200:22 UDP
…

This is probably a host with a remote access program, called PCAnywhere, that is scanning its subnet for PCAnywhere agents. This is known to happen on certain versions of PCAnywhere that are configured, by default, with the NETWORK option. In this case, MY.NET.5.37 scans MY.NET.5.x. The three parts of this signature are local class C scanning and UPD port 5632 and 22. If you need PCAnywhere on this host, then I suggest you disable the NETWORK option.
[PCAnywhere scanning its subnet]

Summary and recommendations

I think that this analysis provides you with better picture of the security of your network. I remind you that it was done with data with a few gaps in coverage and limited time. Nevertheless I believe that I gave you an accurate assessment of the problems that you have and will briefly list what should be done to improve the security of you network infrastructure.

Let me summarize some of the security issues identified in the analysis: You have some compromised hosts which need to be taken care of and investigated immediately. You should report those incidents to your computer incidents response team (CIRT). I would not be surprised that further analysis of the data I was provided with could identify a few more compromised hosts. You are subjected to a lot of port scans, stealth and noisy, done for information gathering purposes. The danger is that vulnerabilities found by these scans will be exploited later by malicious sources. You have too much traffic involving outside IP addresses which are suspected or known to carry malicious activities (watch lists). You have Napster clients running on some of your hosts. You have hosts that are exposed to being exploited and that can be made more secure easily, e.g. changing the default password used on some SNMP devices.

I now want to discuss measures that I think should be taken to improve the security of your network.

You should start by making sure that standard, good practice, procedures are part of your network operations. These would normally include the following:

- Stronger password (change default values),
- Better access control at the perimeter devices, like routers and firewalls,
- Systematically blocking traffic from all IP addresses that you know for sure have no business contacting your inside hosts,
- Tighter control of certain types of traffic, e.g. Telnet and FTP. Consider using encryption-based solutions if you need to offer remote access from outside,
- Etc…

You should implement, or revised, an Acceptable Use Policy that addresses issues like the use of Napster and PCAnywhere.

Your computers, especially the hosts critical to your operation, should have updated patches to avoid being exploited (ex: wu-ftpd exploit). There are vulnerability assessment tools that could help you identify the exposed systems.

I also recommend the strategic deployment of permanent Intrusion Detection Systems (IDS) on your network infrastructure. There are free IDS software products, like Shadow and Snort, which allow very affordable solutions. There are also many popular commercial IDS products like Cisco Secure IDS, ISS Real Secure, NFR and Dragon. If possible, deploy IDS on both sides of your firewalls. This is an excellent way to control the common firewall rule errors issues. Firewall rules are sometime enabled but not systematically verified for correct operations. Your busy firewall system administrators would appreciate the increased confidence in perimeter controls.

I am very interested in being part of your network security solution. I could participate to many phases of a project to significantly improve the security posture of your network, including the Threat and Risk Assessment (TRA) study, the vulnerability assessment study, the selection and deployment of IDS, and the analysis support required to operate the IDS.

Marc Grégoire
GIAC CIA Candidate

Assignment 4 – Analysis Process

This assignment is to describe the process used in analyzing the data in assignment 3. Some of this is included in assignment 3, because I made an effort to explain the steps taken in the Detailed Analysis section. I took the approach of starting from a specific Snort alert, and then progress through my reasoning, telling what the next question should be and how it could be answered, including additional traces where necessary to support my statements. Hopefully, the reader would feel more confident about my conclusion on that alert. I especially tried to verbose for the false positive cases because I believe that carefully discarding false alarms is a big part of the work of the intrusion detection analyst.

Methodology

I list here, in bullet form, the steps of the high level approach I used when confronted to the task of analyzing the scenario.

1) Collect all the material required for the task:

- Practical assignment description (what to do)
- Snort data (www.sans.org)
- Snort program information to understand output and rule creation
- Standard sets of Snort rule (from www.snort.org.)
- TCP/IP protocol reference
- Port lists (Well known ports, registered ports, trojan ports)
- Sample of practical exams from GIAC certified analysts
- Unix and Windows computers
- Web bookmarks for:
  - security sites describing incidents and known attacks (GIAC, NetworkICE, etc…)

    - a site to do nslookup and whois (http://name.space.xs2.net/search/)
- Computer books (Unix, shell scripts, Perl)
- Intrusion detection reference books
    - SANS Parliament Hill 2000 course notes
    - *Network Intrusion Detection – An analyst's handbook*, by Stephen Northcutt.
- Support from spouse (for evenings to be spent working on practical exam)

2) High level statistics on the Snort data.

I found the amount of data overwhelming at first. Because the Snort report data is plain text, with fairly constant patterns, and with one message per line, it lends itself well to be processed with Perl scripts. I figured that other intrusion detection analysts were had faced with analyzing large amount Snort data before. Browsing the Snort web site revealed a few Perl scripts to do statistics. I used a script called *snort_stat*, by Yen-Ming Chen, that I modified to read the format of the Snort files provided for the practical exam.

I merged all the Snort Alert reports into one big file. Then I ran my version of snort_stat.pl on the big alert report file. This gave me 29 pages of sorted statistics. The output lists are:

- The number of attack from same host to same destination using same method,
- Percentage and number of attacks from a host to a destination
- The percentage and number of attacks to one certain host
- Percentage and number of attacks from one host to any with same method
- The distribution of attack methods

The output of snort_stat.pl gave me a good indication of the suspicious activity on MY.NET.

I also extracted all the "spp_portscan" messages from the alert reports because they contain more information than reported by snort_stal.pl. In particular, this line with the total values:

[**] spp_portscan: End of portscan from 208.160.10.19 (TOTAL HOSTS:8 TCP:6 UDP:2) [**]

The list of all the "End of portscan" lines were imported in Microsoft Excel. It is then easy to apply filters, to sort, etc…

I merged all the Snort Scan reports into one big file also. I used many grep commands, or even Windows FIND when I am stock on my Win98 machine at home, and simple shell scripts to extract information from this big scan report file, like:

- How many times each MY.NET hosts is hit by a port scan
- How many hits for each of the 65535 ports

3) High level analysis of the Snort data

I used the top part of the sorted statistics tables just described to establish:

- What are the main attack methods used and that I should research and understand
- What are the main sources of attacks
- What are the most targeted hosts
- What are the most scanned port numbers

I also researched the main IP addresses attacking with whois and nslookup queries.

I treated all the events from two aspects:

- Snort alert event or scan event
- Source is outside or inside MY.NET

4) Detailed analysis of specific Snort alerts

I started analyzing specific alerts from the top of my statistics tables. The analysis steps used were:

- What is really meant by this alert? Research/analyze:
  - Protocol
  - Port numbers involved
  - Snort rule that triggered the alert
  - Correlation: reports on similar attacks (from GIAC and other security/hacker web sites)
- Get a history picture for the source
  - Extract all lines with same IP (as source OR destination)
- Get a history picture for the destination
  - Extract all lines with same IP (as source OR destination)
- If it is a scan, extract all scan traffic with same source from big Snort scan report (visual inspection of this extracted data is very informative)
- Assess the event
  - Inconclusive, if more research needed then what data is required,
  - False positive, (ideally, should rule be modified?)
  - Positive, evaluate the severity

To create a historical snapshot of a specific host I wrote a short batch which I invoke with the IP address as the argument.

5) Tools

My favorite tools, which I used with in the process I described were:

- grep
- Perl
- Unix shell scripting
- Microsoft Excel spreadsheet
- Servant Salamander, a Windows based freeware file manager with built-in viewers and archivers (zip, tar, gz, etc…), excellent for working with Unix files in Windows environment.
- Copernic, Windows interface to multiple search engines, saves results of Web searches.
