Global Information Assurance Certification Paper

SANS GIAC
Intrusion Detection Assignments
Darling Harbour 2001

Assignment 1 - Network Detects

Detect 1: Adore Worm

Apr 6 20:08:16 waterfall.home.org.au ipmon[1348]: [ID 702911 local0.warning] 20:08:15.719774 ipdp0 @200:1 b 136.186.74.253,3157 -> 256.256.99.58,53 PR tcp len 20 60 -S IN

Apr 6 20:08:19 waterfall.home.org.au ipmon[1348]: [ID 702911 local0.warning] 20:08:18.693921 ipdp0 @200:1 b 136.186.74.253,3157 -> 256.256.99.58,53 PR tcp len 20 60 -S IN

1). Source of trace:
My home dialup gateway box.

2). Detect was generated by:
IPFilters 3.4.14 running on Solaris 8 (x86) which logged then dropped the packet.

3). Probability the source address was spoofed:
The packets above are typically used to establish a TCP session, indicating that the source IP address has not been spoofed. The small number packets supports this, therefore the likelihood of spoofing is low.

4). Description of attack:
The above attack was targeting DNS services by initially establishing a TCP session with port 53, which were subsequently discarded. The attempted connections came from a Redhat linux box within an Australian University.

5). Attack mechanism:
Due to the terse nature of the above IPFilter alerts it is hard to determine whether the above was an actual attack or just a stray DNS request. However, the above protocol of the packets (ie TCP) rule out the stray DNS query theory (the TCP side of DNS is typically used for zone transfers and not queries); and the number of packets sent (ie 2) from what was later determined to be a redhat linux box, was too few for a legitimate attempted TCP connection.
I believe the two packets were the first part of an attempted compromise of the system (see below: Correlation Section) by the Adore worm http://www.sans.org/y2k/adore.htm . This variant of the Adore worm uses the TSIG buffer overflow in Bind http://www.kb.cert.org/vuls/id/196945 to compromise a host. The process it uses to compromise the host is as follows:

attacker:port -> victim:53 TCP SYN
victim:53 -> attacker:port TCP SYN ACK
attacker:port -> victim:53 TCP ACK (TCP session established)

attacker:port -> victim:53 UDP DNS inverse query request

where the UDP packet is crafted to exploit the BIND information leak vulnerability http://www.cert.org/incident_notes/IN-2001-03.html .

6). Correlation:

The detect was made on the 6th of April when the Adore worm variant using the Bind TSIG buffer overflow was quite active. The CERT Incident Note IN-2001-03 description of the attack process (ie attacker first tries to establish a DNS TCP session with the victim) correlates with the suspect packets detected. An finger print of the attacking host revealed it was a Redhat Linux box(the target and host of the Adore worm).

7). Evidence active targeting:

As the dialup gateway is only connected to the internet on an "as needs" basis, is allocated a dynamic IP on connection, does not provide DNS services, is not known public to provide DNS services, and the way the Adore worm behaves (ie trawls through networks), I do not believe this host was being actively targeted.

8). Severity:

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

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

= **-3**

system sufficiently safeguarded

where

System Criticality = 0

DNS/Named service is not running

Attack Lethality = 5

Perceived goal was to exploit a buffer overflow to gain root access

System Countermeasures = 4

System regularly patched and Services regularly maintained

Network Countermeasures = 4

External initiated connections denied by regularly maintained firewall

9). Defensive recommendations:

System is sufficiently protected.

10). Multiple choice question:

If the above packets were repeated a few hundred times a second, would it most likely be:

a). a very active Adore worm

b). an attempt DOS by the source IP

c). part of a DDOS of the source IP

d). misconfigured DNS replica

Answer: **C**

Detect 2: Lion Worm

[**] IDS28 - PING NMAP TCP [**]
03/16-11:19:13.207731 194.133.58.129:80 -> 256.256.86.20:53
TCP TTL:49 TOS:0x0 ID:34944
A* Seq: 0x233 Ack: 0x0 Win: 0x578

10:19:13.207573 194.133.58.129.55 > 256.256.86.20.37852: [udp sum ok] udp 10 (ttl 49, id 34942, len 38)
10:19:13.207645 194.133.58.129 > 256.256.86.20: icmp: echo request (ttl 49, id 34943, len 38)
10:19:13.207731 194.133.58.129.80 > 256.256.86.20.53: . [tcp sum ok] 563:563(0) ack 0 win 1400 (ttl 49, id 34944, len 40)
10:19:13.207803 194.133.58.129.53 > 256.256.86.20.53: S [tcp sum ok] 170584361:170584361(0) win 1400 (ttl 49, id 34945, len 40)
10:19:13.220287 256.256.86.20 > 194.133.58.129: icmp: echo reply (DF) (ttl 254, id 0, len 38)
10:19:13.220790 256.256.86.20.53 > 194.133.58.129.80: R [tcp sum ok] 0:0(0) win 0 (DF) (ttl 254, id 0, len 40)
10:19:13.226839 256.256.86.20.53 > 194.133.58.129.53: S [tcp sum ok] 1842276561:1842276561(0) ack 170584362 win 5840 <mss 1460> (DF) (ttl 63, id 0, len 44)
10:19:13.590538 194.133.58.129.53 > 256.256.86.20.53: R [tcp sum ok] 170584362:170584362(0) win 1400 (ttl 49, id 34957, len 40)
10:19:13.590666 194.133.58.129.53 > 256.256.86.20.53: R [tcp sum ok] 170584362:170584362(0) win 1400 (ttl 49, id 34959, len 40)

[**] IDS28 - PING NMAP TCP [**]
03/17-14:53:41.560074 194.133.58.129:80 -> 256.256.86.20:53
TCP TTL:49 TOS:0x0 ID:53535
A* Seq: 0x161 Ack: 0x0 Win: 0x578

13:53:41.559756 194.133.58.129.55 > 256.256.86.20.37852: [udp sum ok] udp 10 (ttl 49, id 53533, len 38)
13:53:41.559826 194.133.58.129 > 256.256.86.20: icmp: echo request (ttl 49, id 53534, len 38)
13:53:41.560074 194.133.58.129.80 > 256.256.86.20.53: . [tcp sum ok] 353:353(0) ack 0 win 1400 (ttl 49, id 53535, len 40)
13:53:41.560144 194.133.58.129.53 > 256.256.86.20.53: S [tcp sum ok] 3400182499:3400182499(0) win 1400 (ttl 49, id 53536, len 40)
13:53:41.572162 256.256.86.20 > 194.133.58.129: icmp: echo reply (DF) (ttl 254, id 0, len 38)
13:53:41.572362 256.256.86.20.53 > 194.133.58.129.80: R [tcp sum ok] 0:0(0) win 0 (DF) (ttl 254, id 0, len 40)
13:53:41.577925 256.256.86.20.53 > 194.133.58.129.53: S [tcp sum ok] 3581211236:3581211236(0) ack 3400182500 win 5840 <mss 1460> (DF) (ttl 63, id 0, len 44)
13:53:41.945305 194.133.58.129.53 > 256.256.86.20.53: R [tcp sum ok] 3400182500:3400182500(0) win 1400 (ttl 49, id 53537, len 40)
13:53:41.949093 194.133.58.129.53 > 256.256.86.20.53: R [tcp sum ok] 3400182500:3400182500(0) win 1400 (ttl 49, id 53539, len 40)

1). Source of trace:
Outside Firewalling and packet filter equipment on external network.

2). Detect was generated by:
Snort on an external IDS provided the snort alerts while Shadow provided the raw network information.

3). Probability the source address was spoofed:
The packets above appear to be trying to determining whether a UDP service is running on port 37852, that the target is reachable, and establishing a TCP session, therefore the likelihood of the source IP address being spoofed is low.

4). Description of attack:
The source starts by sending a group of 4 reconnaissance packets together, with, I believe, the

following purpose:

Packet 1: UDP packet to port 37852. Probably to check for a backdoor

Packet 2: Echo Request. Maybe checking that the host is reachable (but why send the other packets at the same time?), though more likely as part of a fingerprinting exercise.

Packet 3: A TCP-ACK packet from port 80 (HTTP) to port 53 (DNS). This triggered the snort rule "IDS28 - PING NMAP TCP" and is probably also part of a fingerprint exercise. The source port of 80 perhaps chosen to try and avoid IDS detection and/or confuse IDS staff (but we know better than that).

Packet 4: A TCP-SYN packet from port 53 (DNS) to port 53 (DNS). Probably to to establish a TCP session to ensure the Named service is running and in preparation for launching an attack against Named via a known exploit.

The destination responds to packets 2,3 and 4 appropriately, while packet 1 is silently ignored as no such service is running.

Based on the response the source receives, it then tears down/aborts the TCP session it was trying to establish, possibly because its reconnaissance showed the target was not vulnerable or of the wrong platform.

I believe this attack was from the linux Lion worm, which would have used a Bind exploit (probably the TSIG exploit http://www.cert.org/incident_notes/IN-2001-03.html as it was popular at the time of the detect), to compromise the system. If the worm had determined that the system was vulnerable, it would have used the Bind TSIG exploit:

attacker:port -> victim:53 TCP SYN
victim:53 -> attacker:port TCP SYN ACK
attacker:port -> victim:53 TCP ACK (TCP session established)
attacker:port -> victim:53 UDP DNS inverse query request

and compromised the system.

5). Attack mechanism:

The above attack was targeting DNS services as indicated by attempting to establishing a TCP session with port 53, whilst also looking for a backdoor to the system on UDP port 37852. An NMAP style fingerprinting was also used in the initial reconnaissance phase to determine whether the host OS was correct for the worm.

6). Correlation:

The date of the attack (mid March), the source IP and the attack pattern:

attacker:55 -> victim:37852 UDP
attacker -> victim Echo Request
attacker:80 -> victim:53 TCP ACK
attacker:53 -> victim:53 TCP SYN

all seem to correlate with two SANS Global Incident Analysis Centre Reports:

http://www.sans.org/y2k/032401-1230.htm
http://www.sans.org/y2k/032301-0915.htm

regarding the Lion worm:

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

7). Evidence active targeting:

The Lion worm typically scans networks looking for systems running Bind, then tries to exploit those systems, however our Shadow logs showed the only traffic to come from the above source IP was directed solely at our Primary DNS server, therefore I believe it was actively targeting this system.

8). Severity:

Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures)
= (4 + 3) - (5 + 1)
= **1**
system sufficiently safeguarded, though packet filtering could reduce affect if compromised (eg backdoor port not reachable).
where
System Criticality = 4
DNS/Named service is running and was our primary DNS server
Attack Lethality = 3
Perceived goal was to exploit a buffer overflow to gain root access, but not disrupt system services
System Countermeasures = 5
System/Services regularly patched and system stripped/hardened
Network Countermeasures = 1
On raw Internet so only IDS their to alert of attacks

9). Defensive recommendations:
System is sufficiently protected, however the use of a packet filter (ether on border routers or on the host itself) could reduce the damage should the system be compromised (eg backdoor(s) not reachable).

10). Multiple choice question:
Which of the following would most likely cause the Lion worm to give up, similarly to the above attack attempt:
a). no response for any of the first 4 packets
b). a response to all packets
c). a response to packet 4 only
d). a response to packet 1 only
Answer: **A, B and D - with B and D showing the system was possibly already compromised!**

Detect 3: IIS UNICODE attack

[15/May/2001:15:40:56] warning (21847): for host 209.15.2.7 trying to GET /scripts..Á../winnt/system32/cmd.exe, send-file reports: can't find /public/http/cgi-bin..Á../winnt/system32/cmd.exe (No such file or directory)
[15/May/2001:15:41:15] warning (21847): for host 209.15.2.7 trying to GET /scripts/Á/winnt/system32/cmd.exe, send-file reports: can't find /public/http/cgi-bin/Á/winnt/system32/cmd.exe (No such file or directory)

209.15.2.7 - - [15/May/2001:15:41:01 +1000] "GET /scripts/..%c0%qf../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0" 404 207 "-"
209.15.2.7 - - [15/May/2001:15:41:05 +1000] "GET /scripts/..%c1%8s../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0" 404 207 "-"
.
...103 more subtly different variants on the above
.
209.15.2.7 - - [15/May/2001:15:42:18 +1000] "GET /_vti_cnf/..%c0%9v../..%c0%9v../..%c0%9v../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0" 404 207 "-"
209.15.2.7 - - [15/May/2001:15:42:20 +1000] "GET /_vti_cnf/..%c1%1c../..%c1%1c../..%c1%1c../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0" 404 207 "-"

1). Source of trace:
Client web server logs.

2). Detect was generated by:
Netscape Enterprise Webserver on Solaris generated the above alerts. The first two alerts were recorded as webserver system errors in the error log, while the following 109 alerts were recorded as attempted accesses in the access log.

3). Probability the source address was spoofed:
The above alerts are normally the result of an established TCP session therefore the likelihood of the source IP being spoofed is extremely low.

4). Description of attack:
The above attack shows various UNICODE shell metacharacter based exploits used to typically compromise IIS 4 and 5 web servers http://www.whitehats.com/info/IDS434 . Then number exploits attempted *109* for the duration of the attack *84 seconds* suggests that some sort of tool was used to try and compromise the system. From the logs it is plain to see that the attackers goal was to execute */winnt/system32/cmd.exe*.

5). Attack mechanism:
The purpose of this attack was to exploit a UNICODE bug in IIS 4 and 5 via the use of metacharacter so that arbitrary code could be executed.

6). Correlation:
Further log analysis showed that this site had launched similar attacks, all designed around IIS exploits, on the 11th, 15th and 18th of May.

209.15.2.7 - - [11/May/2001:06:28:37 +1000] "GET /_vti_cnf/..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0" 404 207 "-"
209.15.2.7 - - [11/May/2001:06:28:18 +1000] "GET /iisadmpwd/..%e0%80%af../..%e0%80%af../..%e0%80%af../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0" 404 207 "-"
and

209.15.2.7 - - [18/May/2001:17:52:47 +1000] "GET
/msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0" 404 207 "-"
209.15.2.7 - - [18/May/2001:17:52:46 +1000] "GET
/scripts/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0" 404 207 "-"

The source IP traced back to a web hosting domain *hosting4u.net*, so it is possible that a one of the boxes they host has been compromised.

7). Evidence active targeting:
I believe that his system was actively targeted as they launched an attack designed to compromise a webserver at a webserver, however their initial reconnaissance was poor as they were trying to use IIS exploits on a Netscape webserver, to execute Windows NT programs from a Solaris system. Further log analysis showed that they launched similar attacks 4 days prior and 3 days later.

8). Severity:

Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures)
= (4 + 0) - (4 + 4)
= **-4**
system sufficiently safeguarded
where
System Criticality = 4
HTTP service is running
Attack Lethality = 0
Attempting exploit to gain administrator privileges, just wrong platform
System Countermeasures = 4
System regularly patched and Services regularly maintained
Network Countermeasures = 4
Firewall only allows specific services to and from web server.

9). Defensive recommendations:
System is sufficiently protected.

10). Multiple choice question:
Could the above attack worked if the system was running a Netscape webserver on Windows NT:
a). yes
b). no
c). possibly
d). Netscape doesn't have a Windows NT webserver
Answer: **B** - the exploit is with IIS's UNICODE handling.

Detect 4: Queso Fingerprint

[**] IDS29 - SCAN-Possible Queso Fingerprint attempt [**]
03/16-09:50:22.922350 256.256.86.20:3639 -> 202.14.256.256:25
TCP TTL:63 TOS:0x0 ID:0 DF
12****S* Seq: 0x1E295F32 Ack: 0x0 Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 22226160 0 NOP WS: 0

[**] IDS29 - SCAN-Possible Queso Fingerprint attempt [**]
03/16-11:39:34.106001 256.256.86.20:3499 -> 202.14.256.256:25
TCP TTL:63 TOS:0x0 ID:0 DF
12****S* Seq: 0xBA6D7CE0 Ack: 0x0 Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 22881269 0 NOP WS: 0

[**] IDS29 - SCAN-Possible Queso Fingerprint attempt [**]
03/16-12:05:52.915208 256.256.86.20:1827 -> 202.14.256.256:25
TCP TTL:63 TOS:0x0 ID:0 DF
12****S* Seq: 0x1D8D772B Ack: 0x0 Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 23039147 0 NOP WS: 0

[**] IDS29 - SCAN-Possible Queso Fingerprint attempt [**]
03/16-13:36:02.197487 256.256.86.20:1281 -> 202.14.256.256:25
TCP TTL:63 TOS:0x0 ID:0 DF
12****S* Seq: 0x7256559A Ack: 0x0 Win: 0x16D0

1). **Source of trace:**
Outside my employer's firewalls.

2). Detect was generated by:
Snort running on an external IDS sensor.

3). Probability the source address was spoofed:
I would like to think so (as the source host was our external SMTP gateway), however the destination address was outside our network and our IDS was local to the source address. Therefore the likelihood is extremely low.

4). Description of attack:
The source host had been actively Queso fingerprinting multiple hosts for the previous two days. The host began Queso probing remote hosts after a system reboot, however IDS and system logs failed to show any suspicious activity directed at the system prior to the reboot.
E-mails were beginning to arrive from administrators of remote networks asking us to explain why our SMTP gateway was scanning their networks, and clients were beginning to complain about being unable to send e-mails to a few remote sites.
Further analysis showed that the SMTP gateway was only Queso probing other SMTP gateways, and that the mail queue was beginning to backi up for a few internet sites (such as the above destination gateway).
The system logs showed that the system had been rebooted by a staff account, and upon questioning that staff member it was found that the system had been rebooted to use a new linux kernel. The new kernel had been configured to use "Quality of Service and Fair Queuing", which uses the two reserved bits within the TCP flags section in the initial TCP SYN packet to negotiate these features with the remote host if it supports these advanced networking features. Upon reconfiguring the kernel without this feature and rebooting, our SMTP gateway stopped triggering false positives for us and remote

networks, and the e-mail backlog began clearing.

5). Attack mechanism:
The above packets generated by our mail relay contained the two reserved bits and the SYN bit set in the TCP flags section of each packet. This is the signature of a Queso style remote server finger printing attempt (http://www.cert.org/incident_notes/IN-98.04.html) used typically as a reconnaissance mechanism for determining the operating system of various targets. The way the target handles the initial packet, and subsequently the structure of the packet returned by the target provides the fingerprint.

6). Correlation:
By correlating that the Queso detects began after the system was rebooted, and that the only service triggering the alerts were the initial TCP packet sent when negotiating a new SMTP connection to a remote SMTP gateway, we were able to safely predict that the above detects were most likely false positives.

7). Evidence active targeting:
Remote sites would have felt we were actively targeting them as the false positive Queso probes were to their SMTP gateway on the SMTP port.

8). Severity:

Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures)
= (5 + 3) - (1 + 3)
= **4**
better testing before implementing
where
System Criticality = 5
Primary SMTP gateway being blocked by remote SMTP gateways
Attack Lethality = 3
Only a small number of remote sites were actually detecting and subsequently blocking on the false positive
System Countermeasures = 1
Change control procedures
Network Countermeasures = 3
IDS to alert of suspicious behaviour

9). Defensive recommendations:
Stronger testing procedures prior to implementing changes to production systems, and two person teams doing any production change to ensure

10). Multiple choice question:
If an administrator adds a service to a production system without sufficiently testing it, should they:
a). have root access revoked
b). be ridiculed by their peers
c). be forced to become an MCSE where such skills are appreciated
d). be sacked
e). be promoted to webmaster
Answer: A & B and possible C & E

Detect 5: Wingate/Socks scan

```
[**] MISC-WinGate-1080-Attempt [**]
03/17-01:41:43.423245 61.151.158.41:3472 -> 256.256.120.1:1080
TCP TTL:44 TOS:0x0 ID:44857
******S* Seq: 0x38457D   Ack: 0x0   Win: 0x1920
TCP Options => MSS: 536 NOP NOP SackOK
[**] MISC-WinGate-1080-Attempt [**]
03/17-01:41:43.427974 61.151.158.41:3473 -> 256.256.120.2:1080
TCP TTL:44 TOS:0x0 ID:45113
******S* Seq: 0x38457E   Ack: 0x0   Win: 0x1920
TCP Options => MSS: 536 NOP NOP SackOK
.
... sequentially through adjacent C classes 256.256.120 and 256.256.121 ...
.
[**] MISC-WinGate-1080-Attempt [**]
03/17-01:42:19.223798 61.151.158.41:3980 -> 256.256.121.253:1080
TCP TTL:44 TOS:0x0 ID:9022
******S* Seq: 0x38D1AC   Ack: 0x0   Win: 0x1920
TCP Options => MSS: 536 NOP NOP SackOK

[**] MISC-WinGate-1080-Attempt [**]
03/17-01:42:19.412258 61.151.158.41:3981 -> 256.256.121.254:1080
TCP TTL:44 TOS:0x0 ID:9790
******S* Seq: 0x38D1DC   Ack: 0x0   Win: 0x1920
TCP Options => MSS: 536 NOP NOP SackOK

00:41:43.423245 61.151.158.41.3472 > 256.256.120.1.1080: S [tcp sum ok] 3687805:3687805(0) win 6432 <mss
536,nop,nop,sackOK> (ttl 44, id 44857, len 48)
00:41:43.427974 61.151.158.41.3473 > 256.256.120.2.1080: S [tcp sum ok] 3687806:3687806(0) win 6432 <mss
536,nop,nop,sackOK> (ttl 44, id 45113, len 48)
.
... sequentially through adjacent C classes 256.256.120 and 256.256.121 ...
.
00:42:19.223798 61.151.158.41.3980 > 256.256.121.253.1080: S [tcp sum ok] 3723692:3723692(0) win 6432 <mss
536,nop,nop,sackOK> (ttl 44, id 9022, len 48)
00:42:19.412258 61.151.158.41.3981 > 256.256.121.254.1080: S [tcp sum ok] 3723740:3723740(0) win 6432 <mss
536,nop,nop,sackOK> (ttl 44, id 9790, len 48)
```

1). Source of trace:
Outside Firewalling and packet filter equipment on external network.

2). Detect was generated by:
Snort on an external IDS provided the snort alerts while Shadow provided the raw network information.

3). Probability the source address was spoofed:
The packets above appear to be part of a scan trying to establish a TCP session and therefore the likelihood of the source IP address being spoofed is low.

4). Description of attack:
The source quickly scanned two adjacent C Class networks (508 addresses in 36 seconds), looking for the Wingate/Socks service. The source port indicates that this was the only network they were targeting (ie starts at 3472 and increments by 1 for each host, up to 3981 for the last host IP in the

second C Class).
Not much more can be said about this attack/scan. Very simple, very obvious scan (for anyone running an IDS), and very affective when they stumble across ISPs and cable modem networks.

5). Attack mechanism:
The above attack was searching for hosts running the Wingate/Socks services as indicated by attempting to establishing a TCP session with port 1080. Typically they would then use such a service to 'bounce' further attacks through any hosts found at a third site they wished to attack.

6). Correlation:
This is a common scan hackers use to try and find hosts that can be used to hide their tracks http://www.whitehats.com/info/IDS175.

7). Evidence active targeting:
This was a quick scan of two adjacent C Class networks so I would say that their was no targeting involved.

8). Severity:

Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures)
= (0 + 3) - (5 + 5)
= **-8**
no threat
where
System Criticality = 0
Wingate/Socks is not running externally, nor is any other known service on port 1080
Attack Lethality = 3
Goal would be to 'bounce' traffic through such hosts to tracing difficult for the end target
System Countermeasures = 5
Service not running on any servers
Network Countermeasures = 5
Firewalls set to drop all such traffic, and IDS to detect such traffic

9). Defensive recommendations:
Ensure firewalls do drop such traffic, and service is not installed on any systems.

10). Multiple choice question:
During such a scan packets with source port 1080 were seen but not going to the original host, because:
a). they were 'bounced' through wingate/socks running on a system that you didn't know about
b). was next available ephemeral source port at the time
c). one of your systems is scanning for wingate too
d). mangled packet
Answer: **B**

Assignment 2 - Describe the State of Intrusion Detection

Host based Intrusion Detection Systems
A Packet Filtering approach

First proposed as a mechanism for screening networks,Packet Filtering has become the basis for any network security product. Today, all security conscious organisations employ packet filtering gateways to protect their network, with the more aware strategically putting in place Network Intrusion Detection Systems. The ever increasing list of Security Threats/Exploits coupled with the exponential growth of network traffic, especially within intranet/office networks, is eroding the effectiveness of IDS systems in medium to large networks.
Host/Server based packet filtering provides a basic IDS mechanism, whilst ensuring only explicit services are accessible and then only by trusted hosts. This design coupled with IDS systems at network bottlenecks (eg routers) and other critical points within an organisation provides broader coverage and increases the basic security of all hosts within an organisation.

The purpose of this paper is to detail how one can improve the security of hosts within their control via the use of a packet filter with a simple ruleset, and how the logs such a design generates can then be used to compliment Intrusion Detection Systems. In large networks, such as Intranets, the shear volumes of traffic are restrictive on the effectiveness of IDSs. By building a hard coating around your soft centred server, you can significantly reduce the risk to your servers whilst minimising the affect if they are compromised, and alerting you of any anomalies.

Various techniques can be employed to increase the general security of your systems:

- hardening the operating system (eg titan http://www.fish.com/~brad/titan/Titan-Docs/index.html)
- removing all non-essential services
- restricting access to services via their configuration files or via products like tcp_wrappers http://uwsg.iu.edu/security/tcp_wrappers.html)

but these often come with significant costs:

- often a time consuming and involved process
- requires a good knowledge of the operating system you are working with
- requires a good knowledge of the application and services you are working with
- can render the system un-maintainable
- *Patch Clusters* and *Service Packs* often undo all your hard work

The last point is particularly important as sites like http://defaced.alldas.de/ highlight, through the nmap scans they provide of defaced servers, where all too often a site has removed all services except web, but a scan shows that recent patching has re-enable another service (eg rpcbind).

It is the above costs and added complexity that limits what systems that are typically secured in such a fashion. By using a host based packet filter approach, the following benefits can be achieved:

- limits what connections can be made to the server and where from
- limits what connections the server can make and where to
- can provide added protection (eg anti-spoofing, funny IP option, fragmentation)
- provides logs of everything outside of the above bounds

with the following costs

- understanding of what services the system provides and who too
- understanding of what services the system needs access to
- understanding of how to write the packet filtering rulesets
- understanding of how to interpret the packet filtering logs
- slightly increased system load due to the packet filter

It should be noted that this approach does not protect you from exploits targeted at services that you allow from trusted hosts, nor will it detect such attacks! You still need to actively maintain such systems, however the risk that *Service Packs* and *Patch Clusters* can pose is significantly reduced. Also, services that use dynamic ports need to be configured to use as small a set of ports as possible.

The packet filtering product I will use as an example is *IPFilters*, though their are a range of free products available:

- IPFilters for *BSD, Solaris, HP-UX & IRIX
- IPChains / IPTables for linux
- SunScreen Lite for Solaris

The aim is to construct a portable ruleset that is easily maintained so that it can scale as your network scales.

IPFilters background

IPFilters is a freeware stateful packet filter and network address translating firewalling package http://coombs.anu.edu.au/~avalon/ written primarily by Darren Reed. The way the filter works is that each packet is checked against the ruleset sequentially (from top to bottom) and tagged as either being allowed or denied (and possibly logged) depending on the last rule to match the packet. The one exception is the *quick* tag. If a matching rule has the *quick* tag, then no more matching is done, and that rule is applied to the packet.

Rules can be grouped using the *head* tag and then subsequent rules are added to the group using the *group* tag. This is handy for grouping things based on direction, interface and protocol.

Finally, there is a *keep state* tag. This rule can be applied to TCP, UDP and ICMP, even though UDP and ICMP are stateless protocols by definition. IPFilters does this for UDP and ICMP packets by keeping a record (briefly) of any allowed UDP/ ICMP packets. If a matching UDP/ICMP packet is returned (before the initial packet is aged out of the state table), it is allowed back through the filter. This avoids writing messy rules to handle reply packets to stateless protocols (eg DNS queries).

Worked Example

Lets say we had an Intranet with the following characteristics:

10.254.254/24	Private Network
10.254.254.1/32	Route to Internet
10.254.254.2/31	Corporate Systems (DNS, SMTP, ...etc...)
10.254.254.101	Web Backend Oracle Server
10.254.254.201	Corporate Web Server
10.254.254.8/29	Administration Systems

plus many more servers that are superfluous to this example. We will focus on the ruleset required for the Corporate Web Server which serves only internal staff and interfaces with an Oracle database backend.

I have broken the IPFilter ruleset into 6 sections. Apart from two lines in section 4 (ie anti-spoofing of the server's IP lines) and all of section 5 (ie host specific services), the rest of the ruleset should be identical for any other server of desktop in the organisation.

Section 0: Default Deny Rule

Like all good security people, our first answer to everything is **NO** (this always simplifies security issues). We do this with the below two lines. Also, we start defining groups. Groups are handy in analysis as the group number is part of logging, therefore pointing us at the subset of rules that could have been triggered.

Section 1: Grouping Alerts

Next we set up some more groups to make our logs a bit more meaningful.

Section 2: Nasty Packets

Lets start by blocking and logging some nasty packets.

a). Anything with a *ttl* of 1. Typically packets with a low *ttl* are being used for reconnoissance, such as the traceroute program. Unfortunately *IPFilters* only accepts explicit *ttl* values and not ranges, otherwise we could specify values of **X or less**.

b). Anything with strange *IP options* set. This will catch packets with strange IP options set.

c). Anything with *fragments*. We should rarely receive packets that have been fragmented, especially within an organisation. We have two types of fragments defined. *short* refers to fragments that are so small/short that they could only truthfully be malicious. *frag* refers to any packets with fragmentation.

Section 3: Loopback Interface

Now we look at the loopback interface. UNIX systems use the loopback interface heavily, so we set up a few anti-spoofing rules before allowing full access to the loopback interface.

Section 4: Anti-Spoofing

Next we look at anti-spoofing rules based on which network(s) are not allowed to connect to this server, and ensuring that this host does not send/receive any ambiguously addressed packets.

Section 5: Host Specific Services

Finally, we enable the various services this host provides and requires. The above anti-spoofing rules limit who can access this system already, however we can further restrict this. We could replace all "*any*" entries in this section with *10.254.254.201/32* as they refer to the host, however the above anti-spoofing rules should have dealt with packets that weren't relevant to this server. This makes the ruleset more portable as the server's IP isn't hard coded throughout the ruleset, without reducing the

quality of the ruleset (provided your anti-spoofing rules are correct).

Section 6: Closing Groups

This section is not required, however it provides us with more useful logs by closing the original groups we created. Packets that pass through to this point do not have the correct group allocated to them as we only created the various groups (based on direction, interface and protocol type) but did not activate them. We do this by using the "group" field where we initially had the "head" field.

Tests

Traceroute from allowed system
```
# traceroute -p 22 10.254.254.201
18:46:47.317919 10.254.254.9.52414 > 10.254.254.201.22:  udp 12 (DF) [ttl 1]
May  3 18:46:47 www ipmon[115]: 18:46:47.317919  le0 @901:1 b 10.254.254.9,52414 ->
10.254.254.201,22 PR udp len 20 40  IN
```

TCP ACK scan on allowed port from allowed system
```
# nmap -sA -p 22 10.254.254.201
18:26:14.611539 10.254.254.9.47821 > 10.254.254.201.22: . ack 0 win 3072 (DF)
May  3 18:26:14 www ipmon[115]: 18:26:14.611539  le0 @201:1 b 10.254.254.9,47821 ->
10.254.254.201,22 PR tcp len 20 40 -A IN
```

Small fragment attack on allowed port from allowed system
```
# nmap -sS -f -p 22 10.254.254.201
18:23:39.527878 10.254.254.9.36736 > 10.254.254.201.22: [|tcp] (DF)
May  3 18:23:39 www ipmon[115]: 18:23:39.527878  le0 @903:1 b 10.254.254.9,36736 ->
10.254.254.201,22 PR tcp len 20 36 -S IN
```

Attempt to access system external to organisation on non-allowed port:
```
# telnet 192.168.1.1 27374
May  3 19:25:10 oscar ipmon[115]: 19:25:09.760392  le0 @250:1 b 10.254.254.201,32774 ->
192.168.1.1,27374 PR tcp len 20 44 -S OUT
```

Example IPFilter ruleset

```
#
##############################################################################
##### SECTION 0 - Default Deny Rules
#
block in   log all head 100
block out log all head 150
#
##############################################################################
##### SECTION 1 - Grouping Alerts
#
block in   log on le0                 all head 200 group 100
block in   log on le0 proto tcp    all head 201 group 200
block in   log on le0 proto udp   all head 202 group 200
block in   log on le0 proto icmp all head 203 group 200
#
block out log on le0                 all head 250 group 150
block out log on le0 proto tcp    all head 251 group 250
block out log on le0 proto udp   all head 252 group 250
block out log on le0 proto icmp all head 253 group 250
#
##############################################################################
##### SECTION 2 - Nasty Packet Handler
#
block in   log          ttl 1 all head  901
block in   log quick ttl 1 all group 901
block out log          ttl 1 all head  951
block out log quick ttl 1 all group 951
#
block in   log          all with ipopts head  902
block in   log quick all with ipopts group 902
block out log          all with ipopts head  952
block out log quick all with ipopts group 952
#
block in   log          all with short  head  903
block in   log quick all with short  group 903
block out log          all with short  head  953
block out log quick all with short  group 953
#
block in   log          all with frag   head  904
block in   log quick all with frag   group 904
block out log          all with frag   head  954
block out log quick all with frag   group 954
#
##############################################################################
##### SECTION 3 - Loopback Interface
#
block in  log quick from 127.0.0.0/8 to any group 200
block in  log quick from any to 127.0.0.0/8 group 250
pass  in  quick on lo0 all
pass  out quick on lo0 all
#
##############################################################################
##### SECTION 4 - Anti-spoofing
#
block in  log quick from !10.254.254.0/24     to any group 200
block in  log quick from 10.254.254.0/32      to any group 200
block in  log quick from 10.254.254.255/32  to any group 200
```

```
block in  log quick from 10.254.254.201/32  to any  group 200
block in  log quick from any to !10.254.254.201/32 group 200
#
block out log quick from any to !10.254.254.0/24     group 250
block out log quick from any to 10.254.254.0/32      group 250
block out log quick from any to 10.254.254.255/32  group 250
block out log quick from any to 10.254.254.201/32  group 250
block out log quick from !10.254.254.201/32 to any group 250
#
##############################################################################
##### SECTION 5 - Host Specific Services
#
#
##### Internal System ONLY, therefore NO interaction with the Internet router required
#
block in   log quick from 10.254.254.1/32      to any group 200
block out log quick from any to 10.254.254.1/32      group 250
#
##### Allow host to perform DNS / LDAP / SMTP / SYSLOG to Corporate Systems
#
pass out quick proto udp from any to 10.254.254.2/31 port = 53 keep state group 252
pass out quick proto tcp  from any to 10.254.254.2/31 port = 389 flags S keep state group 251
pass out quick proto tcp  from any to 10.254.254.2/31 port = 25 flags S keep state group 251
pass out quick proto udp from any to 10.254.254.2/31 port = 514 keep state group 252
#
##### Allow SSH from Administration Systems to host
#
pass in  quick proto tcp  from 10.254.254.8/29 to any port = 22 flags S keep state group 201
#
##### Allow Ping / HTTP / HTTPS from Private Network to host
#
pass in  quick proto icmp from 10.254.254.0/24 to any icmp-type echo keep state group 203
pass in  quick proto tcp    from 10.254.254.0/24 to any port = 80 flags S keep state group 201
pass in  quick proto tcp    from 10.254.254.0/24 to any port = 443 flags S keep state group 201
#
##### Allow SQL*NET from host to Oracle Server
#
pass out quick proto tcp from any to 10.254.254.101/32 port = 1521 flags S keep state group 251
#
##############################################################################
##### SECTION 6 - Closing Groups
#
block in   log quick on le0               all group 200
block in   log quick on le0 proto tcp    all group 201
block in   log quick on le0 proto udp   all group 202
block in   log quick on le0 proto icmp all group 203
#
block out log quick on le0                 all group 250
block out log quick on le0 proto tcp    all group 251
block out log quick on le0 proto udp   all group 252
block out log quick on le0 proto icmp all group 253
#
##############################################################################
```

Assignment 3 - Analyse This

Introduction
The *Fundamentalist Internet Extremists Pty Ltd* has undertaken to provide a detailed analysis of the network security of GIAC University. A large set of data files containing collected network traffic and alerts were supplied. Unfortunately, no network diagrams were supplied, nor any information on what purpose various hosts and networks served within GIAC University. This made analysis more time consuming, however it ensured that there were no misconceptions on what would be perceived legitimate traffic.

Method
All logs were processed using standard UNIX commands and ad-hoc UNIX and Perl scripts.

Key fields were identified within the logs and files contain subsets of data, based on these fields and other identifying information, were created.

Summaries were then derived from the various subsets of data, and statistics were generated.

Finally, the information generated was correlated to look for various patterns, and Internet sites were referenced to help identify all anomalies.

The following information is a by product of the above procedures, and is the basis for the recommendations made.

The Logs

Below is a summary of logs provided. The logs were named in such a fashion that you could distinguish what type of data they held (A for Alert, S for Scan, and OOS for Out of Spec), however the numbering used didn't aid in identifying the period of the log. Also there were some duplicate logs as indicated by the numbers in the table below. Finally, it is quite evident that there are some serious flaws in the collection of logs as indicated by the vast number of missing logs.

Date	Alerts	Scans	OOS		Date	Alerts	Scans	OOS		Date	Alerts	Scans	OOS
					20001201	1				20010101	1	2	
					20001202	1				20010102	1	1	
					20001203	1				20010103	1	1	
					20001204	1				20010104	1		1
					20001205	1	1			20010105	1		1
					20001206	1	1			20010106	1		
					20001207	1	1			20010107	1		
					20001208	1	1			20010108	1	1	1
					20001209	1	1	1		20010109	1	1	1
					20001210	1	1	1		20010110	1		1
					20001211					20010111	1	1	1
					20001212	1	1	1		20010112	1	1	1
					20001213	1	1	1		20010113	1	1	1
					20001214					20010114			1
					20001215	1		1		20010115	1	1	1
					20001216	1	1			20010116	1		1
					20001217	1	1			20010117			1
					20001218					20010118	1		1
					20001219			1					
					20001220	1	1	1					
					20001221	1	3						
					20001222	1							
					20001223	1							
20001124	1				20001224	1	1						
20001125					20001225		1						
20001126	1				20001226	1							
20001127					20001227		1						
20001128	1		1		20001228	1	1	1					
20001129	1				20001229	1	1						
20001130					20001230	1	1						
					20001231	1	1						

Alerts

Below is a summary of the Snort Alerts provided. I will provide a brief explanation of each alert, and highlight any concerns relating to the alerts themselves. One interesting point - the bracketed field in the totals section shows how many of that column's total was the *MY.NET* address. This is disturbing as it shows a number of alerts were triggered by hosts within *MY.NET*, going to both remote and local hosts.

Snort Signatures	**# Alerts**	**# Source IPs**	**# Dest IPs**
Watchlist 000220 IL-ISDNNET-990517	105918	46	100
SYN-FIN scan	51192	37	27067
DNS udp DoS attack described on unisog	16146	8	6
Tiny Fragments - Possible Hostile Activity	5340	27	13
connect to 515 from outside	4238	10	2877
Watchlist 000222 NET-NCFC	2401	31	19
WinGate 1080 Attempt	2239	474	572
Attempted Sun RPC high port access	2053	16	23
Null scan	826	527	173
Queso fingerprint	710	52	72
SNMP public access	591	20	7
NMAP TCP ping	558	47	156
Russia Dynamo - SANS Flash 28-jul-00	546	2	2
SMB Name Wildcard	515	93	171
SUNRPC highport access	204	25	19
connect to 515 from inside	159	10	98
Broadcast Ping to subnet 70	154	24	1
TCP SMTP Source Port traffic	100	5	88
Back Orifice	77	10	71
External RPC call	59	15	25
Probable NMAP fingerprint attempt	8	5	6
site exec - Possible wu-ftpd exploit - GIAC000623	2	2	2
STATDX UDP attack	1	1	1
SITE EXEC - Possible wu-ftpd exploit - GIAC000623	1	1	1
Happy 99 Virus	1	1	1
TOTALS (*MY.NET* only TOTAL)	**194039 (892)**	**1467 (31)**	**27727 (27715)**

Alerts in Detail

I have broken the alerts into 3 groups:

1. Critical - those which show signs of a system acting in manner that would suggest it has been compromised, or use of a system in a malicious manner (eg DDoS, SPAM).
2. Dangerous - those alerts which show an attempt to compromise a system
3. Warning - those alerts that are primarily related to information gathering, which is often a precursor to an attack

1). Critical

Connections to 515

This port is typically for network printing. There are a number of know exploits for this services on various platforms (http://www.whitehats.com/info/IDS456, http://www.whitehats.com/info/IDS457) and some DoS.

The hosts 209.217.166.69 and 141.211.176.99 trawled through parts of the MY.NET network, while the host 216.119.15.88 seemed to target:

MY.NET.214.166 with 207 alerts
MY.NET.130.86 with 258 alerts
MY.NET.99.104 with 403 alerts
MY.NET.100.209 with 405 alerts

Also, the hosts 62.46.70.175 and 192.118.36.9 showed definite packet crafting as they sent packets with source ports less than 1023.

More suspicious was the traffic generated from MY.NET. The host **MY.NET.70.38** is of particular concern as it scanned a number of hosts in the MY.NET.0 network and the external host 212.187.65.135. Seven other MY.NET hosts sent the odd LPD packet to remote sites.

Tiny Fragments - Possible Hostile Activity

Tiny fragments are typically used for stealthy reconnaissance (as all the fragments have to collected and put together to get a full picture of what is going on), and/or as a DoS (you can craft fragmented packets so that anything trying to rebuild them needs to allocate a large amounts of memory), as they do not typically occur naturally in the wild. Unfortunately, this only alerts us to the possibility of a scan or DoS. If the raw packets were available, a more through analysis could be done.

Two suspiciously looking IPs:

4.4.4.4
8.8.8.8

launched a total of 168 tiny fragments at MY.NET.60.11 around the same time. These IPs look crafted, so their purpose was probably as a DoS against MY.NET.60.11.

Finally, on 20001129, **MY.NET.219.122** was detected as sending a number of tiny fragments at 208.162.62.208. This is particularly disturbing as it indicates that this host could be compromised.

External RPC call

Remote Procedure Calls are one of the most effect ways of compromising a system, by allowing the remote execution code. A number of hosts in the MY.NET network appeared to have been probe for

the services, however the host **MY.NET.6.15** drew repeated attention from a number of external hosts and should be suspected of being compromised.

STATDX UDP attack

This alert informs us someone tried to use the *STATD* exploit to compromise a system http://www.whitehats.com/info/IDS442. This alert does not say if the attempt was successful, but the host launching the attack 206.210.80.6, at **MY.NET.6.15**, ran a number of tests against *rpcbind* prior to trying *STATD* exploit. It would be worth checking this thoroughly to determine if it has been comprised.

SUNRPC highport access and attempted access

A number of alerts concerning port 32771 were detected. Some of these are false positives as 32771 is a legitimate ephemeral port, some alerts appeared to be legitimate external RPC communications (if there is such a thing), however multiple hosts in the 205.188.153/24 network attacked a number of MY.NET hosts. In all cases the source port remained 4000, suggesting packet craft. I would treat all the below hosts as potentially compromised:

MY.NET.213.158	554 alerts detected
MY.NET.222.218	434 alerts detected
MY.NET.97.213	224 alerts detected
MY.NET.223.106	221 alerts detected
MY.NET.224.138	214 alerts detected
MY.NET.105.115	90 alerts detected
MY.NET.97.208	78 alerts detected
MY.NET.98.238	57 alerts detected
MY.NET.97.96	43 alerts detected
MY.NET.226.242	14 alerts detected
MY.NET.97.245	12 alerts detected
MY.NET.224.62	11 alerts detected
MY.NET.98.192	10 alerts detected
MY.NET.97.45	10 alerts detected
MY.NET.220.118	8 alerts detected
MY.NET.98.226	7 alerts detected
MY.NET.97.74	6 alerts detected
MY.NET.97.163	3 alerts detected
MY.NET.225.234	3 alerts detected

DNS udp DoS attack described on unisog

This is a DDoS targeting the IP address B (see http://www.sans.org/y2k/011101.htm). What I assume are your three DNS servers

MY.NET.1.3
MY.NET.1.4
MY.NET.1.5

were used as the repeaters for the DDoS, which generated around 16132 packets over a 90 minute period, starting at 18:30-EST 20010106.

Watchlist 000220 IL-ISDNNET-990517

Their were vast amounts off alerts from the Israel network 212.179/16 and MY.NET. A large portion of this traffic look to be recreational (eg Napster and Online Gaming), however we can not be 100% sure of covert communications over known services (eg 6688 and 6699 for Napster) is not happening. The host 212.179.58.12 seemed to have an active telnet session to **MY.NET.60.11** for over 30 minutes, which requires further investigation.

Watchlist 000222 NET-NCFC

This alert references all traffic coming from the 159.226/16 network in China to the MY.NET network. Because it doesn't look at the traffic in the opposite direction, it can be misleading. The activity recorded falls into three groups:

a). a number of sites in this network have been scan for SMTP relays in MY.NET, and possibly relaying mail through any gateways found. The following sites account for almost half of the alerts, and should be looked at to ensure they are not being used as mail relays:

MY.NET.100.230	737 alerts to port 25 (SMTP)
MY.NET.253.41	264 alerts to port 25 (SMTP)
MY.NET.253.42	103 alerts to port 25 (SMTP)
MY.NET.253.43	100 alerts to port 25 (SMTP)

b). the host 159.226.121.37 appeared to be using various services from MY.NET servers (IMAP2, SMTP and SSL). Unless you have a valid user work in this network, you should definitely examine this. The connection were:

MY.NET.6.7	505 alerts to port 143 (IMAP2)
MY.NET.5.29	275 alerts to port 443 (SSL)
MY.NET.253.53	57 alerts to port 25 (SMTP)
MY.NET.253.51	39 alerts to port 25 (SMTP)
MY.NET.253.52	18 alerts to port 25 (SMTP)

c). the host 159.226.47.14 makes periodic ftp connections to MY.NET.145.18, but doesn't seem to do anything (ie no data connections).

2). Dangerous

Site Exec - Possible wu-ftpd exploit
This alert is in response to someone trying to execute a command on the remote ftp server http://www.whitehats.com/info/IDS317. This has the potential to compromise the destination and gain root access. The three hosts that this was detected for were:

MY.NET.156.127:21
MY.NET.130.98:21
MY.NET.97.162:21

These hosts need to be reviewed to determine if they are running an exploitable version of wu-ftpd and assessed on whether they truly need to provide ftp services to the internet community.

Happy 99 Virus
An e-mail was delivered to MY.NET.6.47 on 20001222 possibly infected with the this virus http://www.cert.org/incident_notes/IN-99-02.html. Every time the infected machine sends and e-mail, a second e-mail is sent with the virus. As no alerts from MY.NET hosts were detected, then hopefully your virus scanning software removed the virus. It is worth checking virus scanning logs to make sure this is the case, and if not track it down through your mail logs.

NMAP TCP Ping
This alert typically is associated with reconnaissance http://www.whitehats.com/info/IDS28. Some UNIX worms have been incorporating such scans as part of their own initial reconnaissance before launching an attack against a site. The *Lion* worm does this http://www.cert.org/incident_notes/IN-2001-03.html, often triggering IDSs with this alert. Any such alerts with a source port of either 80 or 53 and a destination port of 53 for DNS servers and 25 for SMTP gateways should be investigated further. Unfortunately, raw TCP logs aren't available and a list of servers was not supplied, so we can only speculate based on traffic.
The following servers were *pinged* on the DNS port 53:

MY.NET.1.3
MY.NET.1.4
MY.NET.1.5
MY.NET.1.8
MY.NET.1.9
MY.NET.1.10

The following servers were *pinged* on the SMTP port 25:

MY.NET.253.42
MY.NET.6.47
MY.NET.253.41
MY.NET.253.43
MY.NET.6.35
MY.NET.6.34
MY.NET.100.230
MY.NET.110.39

Broadcast Ping to subnet 70
Pings to the broadcast of any network will normally be replied to by most equipment on the network,

with the exception of Microsoft Windows machines. There are two goals of such traffic:

1). to map what equipment you have in a network
2). to use in a DDoS/Smurf attack

and there is evidence of both, with the most serious being someone using this network to perform a smurf DoS on the primary nameserver for *endzone.ro* (213.154.131.131) on 20001201.

Russia Dynamo - SANS Flash 28-jul-00

On the surface this looks like a host in Russia 194.87.6.38 establishing a Napster session with your host MY.NET.205.138 on 20001208 for about 30 minutes. You have created a special rule for this network, so I would recommend further investigation.

3). Warnings

SYN-FIN scan
This type of crafted packet scan is designed to detect open ports and is often used to either trawl through networks targeting a specific port or target a single host and find all TCP services running on it. It was considered a stealth type scan with the possibility of penetrating firewalls, however IDSs and stateful firewalls detect this activity. This sort of activity is typically used for reconnaissance, prior to targeting specific services on those hosts that respond. The majority of these alerts concerned three hosts:

211.34.40.1 with 17604 alerts
195.56.182.206 with 9878 alerts
194.234.48.26 with 8565 alerts

scanning large sections of the *MY.NET* network on ports

53 (DNS) with 18862 alerts
21 (FTP) with 21604 alerts
109 (POP-2) with 9099 alerts

A slower scan (over a month period) of host MY.NET.253.112 from multiple sites was noticed. The 18 detects always had a source port of 32808 and a dest port of 259!

WinGate 1080 Attempt
The Wingate service is used to share internet connections and is therefore highly sought after as it allows people to relay through them, thus hiding their true location http://www.whitehats.com/info/IDS175. These alerts appeared to be scans of various systems.

Null scan
These are packets with no flags set and are often used for stealth scanning. These are crafted packets, and a number of the alerts show strong packet crafting (eg source and destination ports of 0).

Queso fingerprint
This is typically used to fingerprint a remote OS http://www.whitehats.com/info/IDS29 by setting the TCP flags "12S". Linux's "Quality of Service" networking feature legitimately sets these flags and subsequently is a cause of many false positives.

SNMP public access
This alerts us to more information gathering attempts, and suggests we should check ALL MY.NET equipment providing public community snmp information to ensure that it is not writeable.

SMB Name Wildcard
This is typically used for reconnaissance as this service provides name service information, however there are some known exploits http://www.cert.org/vul_notes/VN-2000-03.html. It is hard from the queries alone to determine malicious activity, however three MY.NET hosts:

MY.NET.202.30
MY.NET.111.156
MY.NET.101.160

queried other MY.NET hosts. This could just have been a nosey user on these machines at the time, and not the machine doing anything malicious, however it would probably be worth speaking to whomever was logged on the machine at the time.

Back Orifice

A number of hosts scanned sections of the MY.NET network for the *Back Orifice* backdoor http://www.whitehats.com/info/IDS188. This is a particularly nasty trojan http://www.cert.org/vul_notes/VN-98.07.backorifice.html. No sustained activity was reported so we can take it that the trojan was not found. It would be nice to go through the raw TCP logs to make sure no server did respond.

TCP SMTP Source Port traffic

This appears to be more scans designed to hopefully avoid detection.

Scans and Out of Spec Summary

Scans themselves are typically used as an information gathering exercise, and are more irritating than harmful, with the following exceptions:

1. when used as a DoS against us
2. when used against us to act as a repeater as part of a DDoS
3. when they are originating from something you control (ie possibly compromised host)

It is the third point that is of most concern to us.

The Top 5 Scans of MY.NET hosts for any one day were:

Source	Destination	Time	Duration	#Alerts	Scan Type
24.4.196.167	MY.NET.223.86	08:14 2000/12/05	~ 25 mins	29528	SYN portscan
24.180.134.156	MY.NET.201.78	05:39 2000/12/06	~ 45mins	24415	SYN and UDP portscans
24.26.40.11	MY.NET.223.86	10:08 2000/12/05	~ 20 mins	18744	SYN portscan
66.20.207.21	MY.NET.98.182	11:34 2000/12/28	~ 30 mins	9262	SYN portscan
216.66.200.242	MY.NET.203.94	21:05 2000/12/30	~ 45mins	7134	SYN and UDP portscans

The Top 5 Scans from MY.NET hosts for any one day were:

Source	Destination	Time	Duration	#Alerts	Scan Type
MY.NET.1.[3-5]	203.164.58.41	02:44 2000/12/24	~ 5mins	6459	clusters of DNS query replies (UDP)
MY.NET.60.16	216.15.60.112	13:23 2000/12/28	~ 5mins	5348	UDP portscan with fixed source port of 1298
MY.NET.202.94	207.46.204.86	04:21 2001/01/01	~13hrs	4297	online gaming over UDP port 9000
MY.NET.217.150	216.3.226.131	00:00 2000/01/15	~6hrs	3842	slow repetitive scan of ports 1788/1799
MY.NET.1.[3-5]	203.18.238.26	05:36 2000/12/24	~ 20mins	6459	clusters of DNS query replies (UDP)

Of more importance though, are those MY.NET hosts that generated suspicious traffic, hinting at that they maybe compromised. The next 3 tables highlights such activity.

Miscellaneous Scans from MY.NET network

Source	Destination	Time	Duration	# Alerts	Scan Type
MY.NET.60.16	216.15.60.112	13:23 2000/12/28	4secs	**5348**	UDP portscan with fixed source port (1298)
MY.NET.70.163	24.3.45.174	15:18 2001/01/03	~ 40mins	**1581**	SYN and UDP portscan
MY.NET.201.210	TCP:59	05:44 2001/01/13	~15mins	**2053**	multihost SYN scan of port 59
MY.NET.98.238	172.147.[5-45]	11:43 2001/01/15	~15mins	**7148**	multihost SYN scan for SubSeven trojan (27374)
MY.NET.70.38	MY.NET.0/24	14:27 2001/01/18	~ 9hrs	**396**	NMAP TCP ping and portscan for UDP:515

Crafted Packet Scans

The following table is a list of MY.NET hosts that are suspected of being compromised as they generated packets that appeared to have crafted flags (eg VECNA 21*F*P** RESERVEDBITS). I have listed only those alerts greater than 9 for any one host on any one day. A total of 57 MY.NET hosts showed packet crafting, though some of these were false positives as they were responding to crafted packets that were sent to them.

Source	Date	# Alerts
MY.NET.217.158	12 Jan 2001	**3124**
MY.NET.217.158	13 Jan 2001	**2702**
MY.NET.217.150	11 Jan 2001	**1918**
MY.NET.217.150	15 Jan 2001	**1518**
MY.NET.219.126	8 Jan 2001	**1185**
MY.NET.217.182	20 Dec 2001	**882**
MY.NET.217.158	11 Jan 2001	**645**
MY.NET.217.126	9 Jan 2001	**478**
MY.NET.219.126	9 Jan 2001	**367**
MY.NET.217.182	16 Dec 2001	**216**
MY.NET.217.182	17 Dec 2001	**125**
MY.NET.217.182	21 Dec 2001	**82**
MY.NET.98.152	15 Jan 2001	**37**
MY.NET.186.16	15 Jan 2001	**35**
MY.NET.98.156	29 Dec 2001	**23**
MY.NET.186.16	11 Jan 2001	**16**
MY.NET.186.17	8 Jan 2001	**14**
MY.NET.201.94	1 Jan 2001	**13**
MY.NET.186.16	9 Jan 2001	**12**
MY.NET.186.17	11 Jan 2001	**11**
MY.NET.186.16	8 Jan 2001	**10**
MY.NET.98.185	12 Jan 2001	**10**

Scans with destination ports of 0 and 2000

The following table is a list of MY.NET hosts that are suspected of being compromised as they generated packets that appeared to have crafted ports, eg

Dec 27 03:14:54 MY.NET.98.177:12130 -> 172.152.114.120:2000 SYN **S*****
Dec 27 03:14:54 MY.NET.98.177:12520 -> 155.239.78.17:2000 SYN **S*****
Dec 27 03:14:54 MY.NET.98.177:0 -> 216.175.99.165:0 UDP
Dec 27 03:14:55 MY.NET.98.177:0 -> 216.175.99.165:0 UDP

I have listed only those alerts greater than 9 for any one host on any one day. A total of 28 MY.NET hosts showed packet crafting. I do not know of any legitimate software that behaves like this, so we must assume this is hostile.

Source	Date	# Alerts
MY.NET.98.177	27 Dec 2001	**9416**
MY.NET.97.208	29 Dec 2001	**4606**
MY.NET.97.165	27 Dec 2001	**3930**
MY.NET.98.140	31 Dec 2001	**3526**
MY.NET.97.176	31 Dec 2001	**3161**
MY.NET.98.106	28 Dec 2001	**1883**
MY.NET.97.208	30 Dec 2001	**1762**
MY.NET.97.41	15 Jan 2001	**1288**
MY.NET.98.168	31 Dec 2001	**1264**
MY.NET.97.203	30 Dec 2001	**763**
MY.NET.98.161	1 Jan 2001	**751**
MY.NET.98.106	27 Dec 2001	**602**
MY.NET.97.206	21 Dec 2001	**499**
MY.NET.98.198	16 Dec 2001	**482**
MY.NET.98.198	17 Dec 2001	**200**
MY.NET.98.130	20 Dec 2001	**194**
MY.NET.97.170	20 Dec 2001	**80**
MY.NET.71.38	8 Dec 2001	**62**
MY.NET.98.156	20 Dec 2001	**15**

Recommendations

The shear volume of data reported highlights a need to look closer at what people are doing within GIAC University (ie non=work related activity), and to GIAC University. Once the "non-work related" traffic of online gaming, chat and music streaming is removed from the logs (assuming that no covert channels were operating over these services), we begin to get a picture that all is not well, and the finer we take our analysis, more GIAC University's hosts appear to be generating suspicious traffic.

I would strongly suggest that GIAC University focus on the below key recommendations:

- **Logs** - these are critical if you are truly serious about detecting any breaches. The first table showed a great number of missing logs, for which we are essentially blind to the goings on on these days. A more robust log management procedure is required, with a datestamp in the name of the logs to make them easier to be identified and to avoid log duplication/overwriting.
- **Network Flight Recorder** - if the funds and disk space are available, it is worth capturing at least the first 68 bytes of every packet that enters and leaves your network, so you can replay any traffic that your IDSs alerted you to. This will help clarify if something is a false positive, information gathering exercise or something more malicious.
- **Acceptable Use Policy** - a large portion of the logs were the result of what is often considered "non-work related" traffic (ie chat, online gaming and napster). This activity needs addressing, and if their is a business case then make the appropriate changes to your IDS rules to properly filter the allow services.
- **Network Segregation** - it is worth segregating your network into functional groups (eg infrastructure segment, desktop segment, public segments ...etc...), and assess what the business requirements are for each group to be accessible from the internet. The result of this analysis should be the basis for put a number of networks being some sort of packet filtering gateway(s).
- **Border Router filters** - stronger filters on your border routers is recommended, especially if a large portion of your equipment to remain essentially available to the internet. The filtering of exploitable services that you do not want external people to have access to (eg RPCs, LPD and WinGate), directed broadcasts and basic anti-spoofing rules will reduce your exposure to be exploited/DoS (or used to exploit/DoS other sites) and reduce the number of alerts detected.
- **Compromised Hosts** - a number of hosts showed suspicious activity and should be checked to ensure they are not compromised. If they are found not to be compromised, then I would investigate who was using the system when it generated any suspicious activity.
- **Vulnerability Scanning** - all hosts that have been available to the internet should have a vulnerability scan (such as NMAP), run against them to determine what services are running on the system. Should become a regular exercise on all externally accessible hosts.
- **Training** - I would strongly recommend that GIAC University looks at send key staff on various Internet Security course. SANS offers great courses in this field http://www.sans.org.

Analysis Tools

The above information was derived through the use of UNIX scripts, perl scripts and adhoc command line tools (eg grep, sed, awk, sort, uniq, wc, head, tail, ...etc...).

Example: Scripts involved in deriving the list of MY.NET hosts that have been sending packets with crafted flags.

We start by running my *summary.sh* script. This script calls my perl script *summary.pl* and executes it against each scan log. It assumes the logs are named in the format

YYYYMMDD

and are located in a directory of a similar name.

```
#!/bin/sh
for i in `ls | grep 200`
do
  cd $i
  mkdir scansrc
  mkdir scansrcport
  mkdir scandest
  mkdir scandestport
  mkdir scantype
  ../summary.pl $i
  cd ..
done
```

The *summary.pl* script takes a scanlog and creates files based on:

source
source port
destination
destination port
flags

in their own subdirectory. Eg: the line

Dec 16 09:59:43 MY.NET.209.162:2110 -> 24.31.9.25:2340 INVALIDACK 2***RPAU RESERVEDBITS

would be put in the files

20001216/scansrc/MY.NET.209.162
20001216/scansrcport/2110
20001216/scandest/24.31.9.25
20001216/scandestport/2340
20001216/scanscan/INVALIDACK 2***RPAU RESERVEDBITS

This is not pretty, but if you have the disk space available, it makes analysis a lot easier and quicker.

```
#!/bin/perl
while (<>) {
  chop;
  ($month,$day,$time,$src,$x,$dest) = split;
  ($x,$scantype) = split "-> $dest ";
  ($srchost,$srcport) = split(':',$src);
  ($desthost,$destport) = split(':',$dest);
  open SCANSRC, ">> scansrc/$srchost";
  open SCANSRCPORT, ">> scansrcport/$srcport";
```

```
  open SCANDEST, ">> scandest/$desthost";
  open SCANDESTPORT, ">> scandestport/$destport";
  open SCANTYPE, ">> scantype/$scantype";
  printf SCANSRC "%s\n",$_;
  printf SCANSRCPORT "%s\n",$_;
  printf SCANDEST "%s\n",$_;
  printf SCANDESTPORT "%s\n",$_;
  printf SCANTYPE "%s\n",$_;
  close SCANTYPE;
  close SCANDESTPORT;
  close SCANDEST;
  close SCANSRCPORT;
  close SCANSRC;
}
```

Finally we create a summary report

```
find . -type f -exec wc -l {} \; > summary.txt
```

Now we can start to look for alerts about crafted flags from the MY.NET network.

We first dynamicly create the script *craft.sh* from our summary file

```
grep scantype summary.txt | grep -v UDP | grep -v "SYN \*\*S\*\*\*\*\*" | grep -v "FIN \*\*\*F\*\*\*\*" | sed
's#^.*\./#grep \"MY\.NET\.\*\-\>\" \"#' | sed 's/$/\"/' >> craft.sh
```

We then execute the script and grab the date and source address, and generate alerts statistics

```
./craft.sh | | awk '{print $1,$2,$4}' | cut -d: -f 1 | sort | uniq -c | sort -r > craft.txt
```

References

Security information
http://www.sans.org/
http://www.whitehats.com/
http://www.cert.org/

Port information
http://www.sans.org/y2k/gaming.htm
http://www.tech-nic.net/html/
http://www.snort.org/

General searching
http://www.google.com/
http://groups.google.com/