Global Information Assurance Certification Paper

Intrusion Detection In Depth
GCIA Practical Assignement - SANS 2001 Washington D.C.
Practical Assignment Version 3.0 January 31, 2002

Justin K. Smith, CCNA

Table of Contents

Section One: GCIA Practical

Discussion of Scanning Technology: Stealth Scans

Introduction

The focus of my paper is stealth scanning. The main objective of stealth scanning is to perform reconnaissance on a computer network while masking your identity and also avoiding the triggering of any IDS alerts on the victim's network. I will discuss two methods of performing stealth scanning. The first method uses a Unix utility called hping2; this tool allows you to perform many packet-crafting functions including a method of port scanning using IP spoofing. The second method I will discuss uses a piece of software called Ghost Port Scan. This also spoofs the attackers source address but requires a sniffer to catch the traffic intended for the idle host. These two techniques are not guarantees of anonymity, but do provide a certain degree of stealth.

Hping2

Hping2 is a powerful Unix utility that allows you to craft some useful packets and to perform many types of network reconnaissance. (Hping2 can be found and downloaded at http://www.hping.org/.) The specific technique I will describe in detail involves spoofing the source IP of the attacking host. The attack involves a minimum of three nodes: the attacking node, the idle node and the victim node. After picking your target or victim node, the next step is to select an idle host. The idle host must be idle or in a near idle state. The idea is to find a host that will yield predictable sequence numbers. By counting the increments you can determine if the host is responding only to your stimulus or to your stimulus and other TCP connections. This is achieved by sending TCP packets to the potential idle host using hping2; use the 'hping2 -r' command to display the sequence number increments.

```
[root@attacker /root]# hping2 -r 172.16.0.1
HPING 172.16.0.1 (eth1 172.16.0.1): NO FLAGS are set, 40 headers + 0
   data bytes
len=46 ip=172.16.0.1 flags=RA seq=0 ttl=128 id=32241 win=0 rtt=2.7 ms
len=46 ip=172.16.0.1 flags=RA seq=1 ttl=128 id=+1 win=0 rtt=0.3 ms
len=46 ip=172.16.0.1 flags=RA seq=2 ttl=128 id=+1 win=0 rtt=0.3 ms
len=46 ip=172.16.0.1 flags=RA seq=3 ttl=128 id=+1 win=0 rtt=0.3 ms
len=46 ip=172.16.0.1 flags=RA seq=4 ttl=128 id=+1 win=0 rtt=0.4 ms
len=46 ip=172.16.0.1 flags=RA seq=5 ttl=128 id=+1 win=0 rtt=0.3 ms

--- 172.16.0.1 hping statistic ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.7/2.7 ms
```

You will notice that the id numbers are incrementing by one; this would indicate a possibly idle host. It is true that some operating systems will not increment on a 1:1 ratio, the solution is to not select such a host as the idle node. By using the –r switch you **are** specifying that the sequence ID's be displayed in appending increments rather than printing the actual sequence numbers. The next step is to send a series of SYN packets to the victim host using the idle host's IP address as the source address. This command looks like this: 'hping2 -a [idle host IP address] -p [port#] -S [victim host IP address]'.

```
[root@attacker /root]# hping2 -a 172.16.0.1 -p 22 -S 10.0.4.2
HPING 10.0.4.2 (eth1 10.0.4.2): S set, 40 headers + 0 data bytes
```

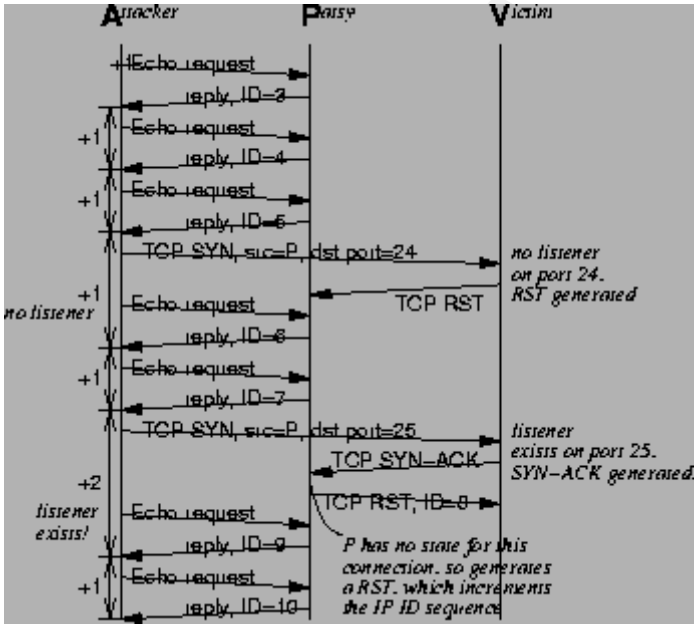
```
--- 10.0.4.2 hping statistic ---
16 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Then in another console window execute the 'hping2 -r' command on the idle node.

```
[root@attacker /root]# hping2 -r 172.16.0.1
HPING 172.16.0.1 (eth1 172.16.0.1): NO FLAGS are set, 40 headers + 0
   data bytes
len=46 ip=172.16.0.1 flags=RA seq=0 ttl=128 id=32329 win=0 rtt=2.6 ms
len=46 ip=172.16.0.1 flags=RA seq=1 ttl=128 id=+2 win=0 rtt=0.4 ms
len=46 ip=172.16.0.1 flags=RA seq=2 ttl=128 id=+2 win=0 rtt=0.3 ms
len=46 ip=172.16.0.1 flags=RA seq=3 ttl=128 id=+2 win=0 rtt=0.3 ms
len=46 ip=172.16.0.1 flags=RA seq=4 ttl=128 id=+2 win=0 rtt=0.3 ms
len=46 ip=172.16.0.1 flags=RA seq=5 ttl=128 id=+2 win=0 rtt=0.3 ms
len=46 ip=172.16.0.1 flags=RA seq=6 ttl=128 id=+2 win=0 rtt=0.4 ms

--- 172.16.0.1 hping statistic ---
7 packets tramitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.7/2.6 ms
```

You can see that the idle host is responding with 'id=+2', this indicates that port 22 on the victim host is a listening port.

Here is a graphic of an Hping2 type of stealth scan:



(http://www.icir.org/vern/papers/norm-usenix-sec-01-html/node8.html)

Hping2 Analysis

I will now discuss the theory of why this attack works and how to further avoid detection. This scan depends on the fact that most operating systems reply to an unsolicited SYN/ACK with a RST, which increments the global packet id number by one digit. However, operating systems ignore unsolicited RST packets, thus the sequence numbers do not increment. Since open ports send SYN/ACK's in response to your SYN and closed ports send RST's to your SYN packet, you can tell if the target host port is open by watching to see if the idle host is sending packets in response to the first stimulus. If the window size increments more than +1 then you have an open port, if it stays at +1 the port is probably closed. To further disguise your reconnaissance you can create a shell script that will scroll through a list of idle hosts for each port it scans. This measure foils some Network Intrusion Detection Systems from seeing a predictable scanning pattern. Also, not picking a repeating sequence of ports will further aid in avoiding detection.

Of course the idle host may be running a NIDS or a firewall and may be logging all of this activity, which could give your true IP away. The SYN/ACK's that will be generated by the victim host may generate an alert on a NIDS and may appear to be backscatter from spoofing activity. Matched with the SYN packets sent from the attacker to the idle host the true scanner could be deduced. However, the low severity of these scans would only peak the interest of a highly paranoid and idle analyst not to mention the difficulty of correlating the two stimuli. No method of reconnaissance can offer 100% anonymity, but this tactic is fairly simple to execute and mostly low risk to the scanner. The backscatter from the victim host to the idle host looks like the capture below:

```
14:54:20.671316 eth1 < victim.http > idle.1910: R 0:0(0) ack
126993058

8 win 0

14:54:21.671316 eth1 < victim.http > idle.mtp: R 0:0(0) ack
1562594610

 win 0

14:54:22.671316 eth1 < victim.http > idle.1912: R 0:0(0) ack
170366927

8 win 0

14:54:23.671316 eth1 < victim.http > idle.1913: R 0:0(0) ack
139309450

9 win 0
```

In this case I was scanning for port 80, the RST flags indicate that the port is not listening on the victim host. The next capture shows a positive response:

```
15:10:05.671316 eth1 < victim.microsoft-ds > idle.2566: S
3872903538:3

872903538(0) ack 1346052189 win 16616 <mss 1460> (DF)
```

```
15:10:06.671316 eth1 < victim.microsoft-ds > idle.2567: S
3873197743:3

873197743(0) ack 135028404 win 16616 <mss 1460> (DF)

15:10:07.671316 eth1 < victim.microsoft-ds > idle.2568: S
3873492416:3

873492416(0) ack 1591602287 win 16616 <mss 1460> (DF)

15:10:08.671316 eth1 < victim.microsoft-ds > idle.2569: S
3873785606:3

873785606(0) ack 1565778464 win 16616 <mss 1460> (DF)
```

This port is listening on port 445 and returns a SYN/ACK. With a little analysis, one could deduce that this is a spoofed packet. The first indication would be the incrementing port number on the idle host. In a normal TCP retransmission situation the port number would probably remain constant. One could also inspect the initial sequence number, which may not be consistent with the operating system of the idle host.

In the next capture you can see a capture made by the victim host:

```
15:55:05.091316   eth1 > idle.1301 > victim.microsoft-ds: S
675014920:675014920(0) win 512

15:55:05.091316   eth1 < idle.1301 > victim.microsoft-ds: S
675014920:675014920(0) win 512

15:55:07.091316   eth1 > idle.1303 > victim.microsoft-ds: S
221348170:221348170(0) win 512

15:55:07.091316   eth1 < idle.1303 > victim.microsoft-ds: S
221348170:221348170(0) win 512
```

In this case the idle host would look like the attacker sending a series of SYN packets attempting to elicit a response from the victim. From the victims point of view it is difficult without data from the idle host to deduce the true source of the attack. The TTL could be examined and if a series of trace routes yields an incompatible number of hops it may be possible to at least determine that this is a spoofed probe. This assumes that the original TTL can be calculated correctly which depends on the attackers operating system and whether the TTL has been manipulated either in the code or in the operating system.

As stated earlier this attack depends on the prediction of TCP sequence numbers. Many operating systems do not make good idle hosts. I have tested this attack on Red Hat Linux 7.1 and on Solaris 8 boxes (with ISN randomization configured '/etc/default/inetinit set to TCP_STRONG_ISS to **2**'), neither of these boxes provided sequence numbers that could be easily predicted or they would not respond to the hping2 reconnaissance. By far

the best hosts are Windows 2000 and Windows NT 4, both provided predictable sequence ID's. I was also able to use SCO Open Server as an idle host. An example of a Windows 2000 server is given below:

```
[root@attacker scripts]# hping2 -S -p 21 10.0.0.1
HPING 10.0.0.1 (eth1 10.0.0.1): S set, 40 headers + 0 data bytes
len=50 ip=10.0.0.1 flags=RA seq=0 ttl=125 id=83 win=0 rtt=1.8 ms
len=50 ip=10.0.0.1 flags=RA seq=1 ttl=125 id=84 win=0 rtt=2.1 ms
len=50 ip=10.0.0.1 flags=RA seq=2 ttl=125 id=85 win=0 rtt=2.0 ms
len=50 ip=10.0.0.1 flags=RA seq=3 ttl=125 id=86 win=0 rtt=1.6 ms

--- 10.0.0.1 hping statistic ---
4 packets tramitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 1.6/1.9/2.1 ms
```

Ghost Port Scan

Ghost Port Scan (GPS) is another Unix based stealth scanner utility that can detect open ports without detection. (GPS can be downloaded at gps.sourceforge.net). GPS uses a combination of IP address spoofing and layer 2 or collision domain sniffing. This tool is a little more difficult to implement. The software sends a SYN packet to the victim with the IP of the idle host as the source address. If the attacker and the idle host are in the same collision domain then GPS will sniff the response off the wire and report the result to stdout. An example of a GPS scan is shown below:

```
[root@attacker docs]# gps -s 172.16.0.1 -p 21-80 -i eth1 -d
10.0.0.2 -S ip
Ghost Port Scan version 0.7.0 by whitehat@altern.org
(gps.sourceforge.net)
Network device: eth1
No open ports found
60 ports scanned
Grabbed 0 matching packets

[root@attacker docs]# gps -s 172.16.0.1 -p 137-139 -i eth1 -d
10.0.0.2 -S ip
Ghost Port Scan version 0.7.0 by whitehat@altern.org
(gps.sourceforge.net)
Network device: eth1
Port 139: netbios-ssn (NETBIOS Session Service) is open
3 ports scanned
Grabbed 1 matching packets
```

The syntax is similar to Hping2, in the above example the –s switch indicates the spoofed source address the –p switch lists the range of ports to be scanned. The –I switch chooses the attackers interface and –d indicates the destination IP address. You will also notice the "ip" statement at the end of the command. This indicates the method that GPS will use to spoof; the default is to spoof the MAC address. In this case we want to spoof the IP

address and let ARP and the router deal with the frame addressing. Spoofing your MAC address can be dangerous and can trigger NIDS within the originating subnet.
With GPS the attacker is again spoofing the source address but instead of also scanning the idle host the attacker waits patiently for the packets to be delivered back to the idle node. If the attacker and the idle host are not in the same layer 2 collision domain then a sniffer must be placed in a collision domain with the idle host. An attacker could also use a port span on a managed switch or use a tool like dsniff if the idle host is on a non-VLAN configured switch. GPS could also be used to spoof and redirect layer 2 frames, however this technique is both potentially destructive and conspicuous. An attacker could also use a tool like sniffit or tcpdump to retrieve the return packets if any to the idle host. This approach would require a box in the same collision domain or the idle host itself to have a covert sniffer installed. Below, is an example using the sniffer Ethereal:

```
Ethernet II
    Destination: 00:50:8b:60:f9:16 (COMPAQ_60:f9:16)
    Source: 00:20:af:f0:f6:92 (3Com_f0:f6:92)
    Type: IP (0x0800)
    Trailer: 000000000000
Internet Protocol, Src Addr: IDLE(172.16.0.1), Dst Addr:
VICTIM(10.0.0.2)
    Source: IDLE(172.16.0.1)
    Destination: VICTIM(10.0.0.2)
Transmission Control Protocol, Src Port: 24288 (24288), Dst
Port: microsoft-ds (445), Seq: 677406330, Ack: 657072
    Source port: 24288 (24288)
    Destination port: microsoft-ds (445)
    Sequence number: 677406330
     Flags: 0x0002 (SYN)


Ethernet II
    Destination: 40:55:7f:3b:06:3f (40:55:7f:3b:06:3f)
    Source: 00:50:8b:60:f9:16 (COMPAQ_60:f9:16)
    Type: IP (0x0800)
    Trailer: 0000
Internet Protocol, Src Addr: VICTIM(10.0.0.2), Dst Addr:
IDLE(172.16.0.1)
    Source: VICTIM(10.0.0.2)
    Destination: IDLE(172.16.0.1)
Transmission Control Protocol, Src Port: microsoft-ds (445), Dst
Port: 24288 (24288), Seq: 389039105, Ack: 677406331
    Source port: microsoft-ds (445)
    Destination port: 24288 (24288)
    Sequence number: 389039105
    Acknowledgement number: 677406331
        Flags: 0x0012 (SYN, ACK)
```

Ghost Port Scan Analysis
This tool shares some common elements with hping2 but also demonstrates some limitations. The idle host must be able to be sniffed to achieve a retrieval of the port

activity information. Also, if the attack is closely analyzed from the LAN the true MAC address of the attacker can be revealed. This presupposes that the attacker and the idle host are contained within one collision domain and that traffic is being sniffed. For example, I tested this attack where the attacker and the idle host were in the same collision domain; I was able to sniff out both the stimulus and the response (as seen in the capture above). The MAC address of the idle host does not match the idle host's IP address. In fact the attacker has given away her true MAC address.

Idle host's MAC = 00:01:02:F3:C3:E0

Attacker's MAC = 00:20:AF: F0:F6: 92

Overall, this attack is clumsier than hping2 but is also more passive if the conditions are correct.

Detection and Defense

The next logical step is to determine how to detect these probes and how to trace it back to its original source. Also, how can you defend against this type of stealth scan. It is fairly simple to detect this type of scan if the scanner does not take precautions to avoid triggering a NIDS. As already stated that the attacker could select multiple idle host's and assign one port to each idle host to avoid a port scan detect on many NIDS. If the attacker took these steps it would be extremely difficult to trace an attacker using hping2 or Ghost Port Scan. Techniques for detecting spoofing could be used, but that would only reveal that the address is being spoofed and would not reveal the attacker's true IP address. The best way to trace this activity would be to coordinate with the owner of the idle host and try to correlate information. This assumes that the idle host is capturing the communication in some fashion and that they would be willing to cooperate.

Another way to defend against a stealth scan that uses specific tools and utilities is it to fingerprint the tools themselves. An example of this type of tactic is a Snort rule that can pinpoint an NMAP scan by the packets that NMAP generates. (NMAP can be downloaded at www.insecure.org). The Snort rule in question can be seen below:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING NMAP";
dsize: 0; itype: 8; reference:arachnids,162; classtype:attempted-
recon; sid:469; rev:1;)
```

A rule could be written to detect certain anomalous feature of either Hping2 or GPS. Hping2 itself can trigger an existing Net Ranger IDS Alert on a Cisco router configured with NIDS. There are ways to increase the stealth of this scan, such as sending a packet with the ACK flag only. This tactic has been used to evade NIDS preprocessors.
The problem with this is that many malformed packets can be used for reconnaissance such as operating system detection but are ill suited to our particular attack. With these types of scans a lack of response is acceptable as it may indicate the type of operating system whereas in spoofed port scanning the response to the idle host must be present.

In terms of defending against this type of activity, it is almost impossible to block all such attacks as a victim. The burden for stopping this behavior is on the administrator of the attacker's and the idle host's network. If the firewall of the attacker's network is configured to not allow spoofing then this activity could be halted. The firewall must be configured to not allow non-internal addresses out to the Internet, thus stopping a host inside the firewall's network from sending a packet out with an incorrect source address. The best way to stop this behavior is to put an end to predictable sequence numbers. Operating systems such as Open BSD have taken cryptological measures to ensure random sequence number generation. Other operating systems such as Solaris 2.6 to 8 have also taken steps but they are not enabled by default. Windows NT 4 is the ideal idle host, although there is a patch that will supposedly solve this problem. There have been some developments in modifying the TCP/IP stack to foil the type of stealth scan seen with Hping2. This tactic involves sending a SYN/ACK right behind every RST. This would yield an increment on the idle host even if the port is not listening. (http://www.icir.org/vern/papers/norm-usenix-sec-01-html/node8.html). There are several methods to foil this type of attack, some on the attackers network and some at the victim and idle host networks. It is ultimately the diligence of network administrators and the efforts of operating system developers that will stop this type of stealthy port scanning.

Conclusion

All of the techniques I have discussed use some degree of stealth in there application. I have enumerated two types of attacks that can hide the true scanners identity. There are also several strategies for stopping this type of activity. From what I have seen as an analyst is that most scanners make little or no effort to disguise their identity. This tactic is suited for the more elite (eleet) attacker who wants to probe for a few specific ports on a particular set of hosts. However, these attacks could be easily automated using scripts and numerous idle hosts. If the TCP/IP stack slowly becomes more secure and firewall and server administrators become more diligent, this attack could be almost totally eliminated.

Section Two: Network Detects

Detect #1: Code Red II Infection Attempts

```
[**] [1:1243:2] WEB-IIS ISAPI .ida attempt [**] [Classification: Web Application Attack] [Priority: 1]
01/15-16:26:42.508067 217.125.118.72:50357 -> 172.16.0.1:80 TCP TTL:240 TOS:0x10 ID:0
IpLen:20 DgmLen:1504 ***AP*** Seq: 0xA2C4FDFA Ack: 0x6AC43140 Win: 0x4470 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS552] [Xref => http://cve.mitre.org/cgi-
bin/cvename.cgi?name=CAN-2000-0071]
```

[**] [1:1243:2] WEB-IIS ISAPI .ida attempt [**] [Classification: Web Application Attack] [Priority: 1] 01/15-17:39:44.328067 24.83.176.243:4214 -> 172.16.0.1:80 TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:2964 ***AP*** Seq: 0xE6BC93AB Ack: 0x245EF02 Win: 0x4470 TcpLen: 20 [Xref => http://www.whitehats.com/info/IDS552] [Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0071]

Source of the trace:
This trace was taken from my employer's corporate web-hosting DMZ. I am running Snort 1.8.3 with a promiscuous and silent NIC sniffing between the firewall and the Internet router.

Detect was generated by:
Snort Intrusion Detection System. I am running Snort Snarf as a tool to convert the alerts into an html format and then I am running a set of custom shell and PERL scripts to maintain and archive the information. This alert was detected with the following Snort rule:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS ISAPI
.ida attempt"; uricontent:".ida?"; nocase; dsize:>239; flags:A+;
reference:arachnids,552; classtype:web-application-attack;
reference:cve,CAN-2000-0071; sid:1243; rev:2;)
```

Probability the source address was spoofed:
From what I was able to learn about this attack it appears unlikely that this attack involved spoofing. However I have noticed that every packet I have detected of this attack in the last few weeks has had a TTL of 240. Perhaps this is a vector parameter; not many operating systems have a default TTL of 255.

Description of attack:
A Windows NT 4 / 2000 server running IIS 4 or 5 that has been infected with the Code Red worm will attempt to propagate to other hosts.

> "There is a **one in two** chance that a given thread will scan random IP addresses with the same first byte as the infected host.
>
> There is a **three in eight** chance that a given thread will scan random IP addresses with the same first two bytes as the infected host.
>
> There is a **one in eight** chance that a given thread will scan random IP addresses."

(http://www.cert.org/incident_notes/IN-2001-09.html)

The CVE number for this attack is cve,CAN-2000-0071.

Attack Mechanism:
This attack makes use of a buffer overflow vulnerability in Windows NT 4.0 (IIS 4.0 and Index Server 2.0) and Windows 2000 (Server and Professional with IIS 5.0). The only precondition for exploiting this vulnerability is that an IIS server is running with script mappings for Internet Data Administration and Internet Data Query; .ida and .idq files

respectively. The Indexing Services do not need to be running. (Microsoft MS01-033). Code Red II uses this buffer overflow to execute code on the victim machine. The Code Red II worm attempts to infect other hosts by issuing an HTTP GET request on port 80 with a specifically crafted packet. This packet attempts to exploit the .ida buffer overflow, thus attempting to infect the target machine. The newly infected machine then begins to look for other vulnerable hosts to infect.

Correlations:
This attack has been well documented by www.cert.org and by many other organizations. A good discussion of this attack can be found at:
http://www.incidents.org/react/code_redII.php

Evidence of active targeting:
This appears to be a random targeting, other than the prerequisite that port 80 is responding.

Severity:
(Criticality + Lethality) - (System + Network Counter Measures) =Severity

(4+5)-(5+1) = 3

Criticality: The servers that were being targeted were in our corporate DMZ so I will assign the criticality to 4. The only factor that prevents this metric from reaching 5 is the fact that we have a good disaster recovery procedure in place.

Lethality: Remote control yields a 5.

System Countermeasures: All of our potentially vulnerable systems were patched to prevent this exploit. The patch provided by Microsoft for this attack is Q300972_W2k_SP3.

Network Countermeasures: We are allowing port 80 traffic to pass through the firewall from the Internet into our DMZ. The only countermeasure is our NIDS server, which at this point does not take any preemptive action against such an attack. This gives us a rating of 1 for network countermeasures.

Defensive recommendation:
As none of our servers were in fact compromised, the only defensive mechanism I would suggest would be perhaps instituting an Application Layer or content filter inside the firewall. Another solution is to install a Network Intrusion Detection System or firewall that takes the initiative of blocking the traffic from the attacking host. I could of course block the traffic at the firewall by making a manual rule, but that strategy could become an administrative problem and may eventually overburden my firewall. I could also use Cisco IOS URL content filtering at the border router to stop this traffic, however the same argument made above could also apply to this solution.

Multiple choice test question:
Which one of the following statements is false?

a) Blocking all port 80 traffic from entering or leaving the DMZ would prevent this attack from being successful.
b) Using a different server operating system and a different web server application could prevent a successful attack by the Code Red II worm.
c) Code Red II can only infect an un-patched Windows 2000, IIS 5.0 server if the server is currently running Indexing Services.
d) If port 80 were left open from the Internet to the corporate DMZ, this attack probably could not have been prevented by using only a port filtering firewall.

Answer: c. A Windows NT/2000 IIS server can be compromised if it is un-patched even if Indexing Services are not running; all that is required is that the binaries be installed and that script mapping is configured with IIS.

[1] http://www.cert.org/incident_notes/IN-2001-09.html
[2] http://www.cert.org/advisories/CA-2001-13.html
[3] http://www.cert.org/incident_notes/IN-2001-09.html
[4] http://www.incidents.org/react/code_redII.php
[5] Microsoft MS01-033

Detect #2: Sadmind Worm Access

[**] [1:1375:1] WEB-MISC sadmind worm access [**] [Classification: Attempted Information Leak] [Priority: 2] 01/15-14:27:25.138067 211.152.65.34:64419 -> 172.16.0.1:80 TCP TTL:232 TOS:0x0 ID:62451 IpLen:20 DgmLen:58 DF ***AP*** Seq: 0x698FCF6F Ack: 0x301B77B3 Win: 0x2238 TcpLen: 20 [Xref => http://"www.cert.org/advisories/CA-2001-11.html"]

[**] [1:1375:1] WEB-MISC sadmind worm access [**] [Classification: Attempted Information Leak] [Priority: 2] 01/16-19:37:30.078067 210.113.136.60:55944 -> 10.0.0.69:80 TCP TTL:235TOS:0x0 ID:32257 IpLen:20 DgmLen:58 DF ***AP*** Seq: 0xE35539DE Ack: 0x28445A78 Win: 0x2238 TcpLen: 20 [Xref => http://"www.cert.org/advisories/CA-2001-11.html"]

[**] [1:1375:1] WEB-MISC sadmind worm access [**] [Classification: Attempted Information Leak] [Priority: 2] 01/16-19:44:04.118067 210.113.136.60:58089 -> 10.0.0.70:80 TCP TTL:235TOS:0x0 ID:33091 IpLen:20 DgmLen:58 DF ***AP*** Seq: 0x438313E0 Ack: 0x2DE0DF60 Win: 0x2238 TcpLen: 20 [Xref => http://"www.cert.org/advisories/CA-2001-11.html"]

Source of the trace:
This trace was taken from my employer's corporate web-hosting DMZ. I am running

Snort 1.8.3 with a promiscuous and silent NIC sniffing between the firewall and the Internet router.

Detect was generated by:
Snort Intrusion Detection System. I am running SnortSnarf as a tool to convert the alerts into an html format and then I am running a set of custom shell and PERL scripts to maintain and archive the information. This alert was detected with the following Snort rule:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-MISC
sadmind worm access"; content:"GET x HTTP/1.0"; offset:0; depth:15;
classtype:attempted-recon; reference:url,"www.cert.org/advisories/CA-
2001-11.html"; sid:1375; rev:1;)
```

Probability the source address was spoofed:
This is unlikely as the two most probable vectors for the spread of this attack involve infected machines that do not attempt to disguise their IP address.

Description of attack:
When a server (Solaris 2.x) is infected with the sadmind worm it attempts to infect other Solaris boxes and also attempts to compromise and deface IIS web servers using a vulnerability with Microsoft's Internet Information Server. This attack can also be seen with the NIMDA Worm attempting to spread itself to IIS servers. The sadmind worm has been categorized as CVE-1999-0977.

Attack Mechanism:
The sadmind attack begins when a Solaris server running an un-patched version of Solstice sadmind. The attacker using a set of buffer overflows is bale to modify the .rhosts file by injecting the string "+ +" thus opening access to the server. A set of PERL and shell scripts are then added to a directory inside the /dev directory and the attack processes then begin. The sadmind worm has two initiatives: it attempts to compromise more Solaris servers and it attempts to deface and compromise Microsoft IIS servers. Infected NIMDA servers also use sadmind to scan for servers that have already been infected by the sadmind worm. NIMDA from what I can deduce appears to looking for the same Directory Traversal Vulnerability as sadmind/IIS. There is the possibility that this attack is being carried out manually in an attempt to deface or DOS the targeted web site.

Correlations:
This attack has been well documented in the Internet community. The attack itself has been designated with the following CVE number: CVE-1999-0977.
This attack has also been well documented by http://www.cert.org/advisories/CA-2001-11.html.

Evidence of active targeting:
A Solaris box infected with the sadmind worm uses a semi-random method for searching

for new hosts to infect. If a NIMDA worm triggered this alert then the vector would probably be more predictable (i.e. first octet… second octet… in common with the victim host). It is also possible that our site was selected by a human attacker who is scanning our server to see if we are susceptible to the directory traversal vulnerability. All of the captures I collected came from East Asia (China and South Korea), this might indicate an attempt to use this attack to deface our websites. This argument is supported by the fact that many web site defacements have attackers that originate in the previously mentioned nations.

Severity:
(Criticality + Lethality) - (System + Network Counter Measures) =Severity

(4+4)-(5+1) = 2

Criticality: These servers are part of our corporate web hosting DMZ yielding a 4.

Lethality: This attack is lethal in the sense that it can deface any existing web pages and can leave a host vulnerable to further attacks. The reason I have not given this attack a 5 is due to the fact that the access an intruder can initially attain is only that of I_USR_COMPUTER_NAME. It is of course possible that an attacker could elevate their permissions once they have some level of user access.

System Countermeasures: The systems that were the target of this attack are all patched for this particular directory traversal vulnerability so I assigned the system countermeasures a 5.

Network Countermeasures: Once again we have to leave port 80 open from the Internet into our DMZ. We can detect most malevolent port 80 traffic using a NIDS, but that is about it for the moment. I have assigned the network countermeasures a 1.

Defensive recommendation:
The only defense for an attack like this is to patch and maintain the web servers. Or we could replace our IIS servers with another platform. Although as stated earlier in the Code Red II discussion we could take certain measures at a gateway, firewall or a preemptive NIDS, the profile for such attacks would take time to develop thus leaving a victim open if their network were an early target. This argument also applies to the applying of patches and host based security strategies. In general with regards to IIS, carefully configuring the permissions of the server such as explicitly restricting the I_USR_COMPUTER_NAME account on all directories outside the /inetpub directory can help to mitigate the damage that could be done by attacks of this type. In one of these attacks I also notified the administrator of the box (which turned out to be a PolyTech High School in South Korea) and notified him that he may be infected with either sadmind if he was running Solaris or NIMDA if it was a Microsoft NT/2000 server.

Multiple choice test question:

Which one of the following statements is true?

a) This worm's initial vector is a Microsoft NT/2000 server that is running Solstice Sadmind.
b) When an IIS server is compromised, the sadmind worm initially gains administrator level access to the server.
c) As long as your company is not running any Solaris servers in the DMZ then you are not vulnerable to this attack.
d) The sadmind worm has two targets Windows NT/2000 IIS servers and Solaris 2.x servers.

Answer: d. The sadmind attack manipulates both Solaris and Windows NT/2000 using a vulnerable Solaris box as a launch pad for further attacks to both other Solaris servers and to Windows NT/2000 IIS web servers.

Detect #3: cc command attempt

[**] [1:1344:1] WEB-ATTACKS cc command attempt [**] [Classification: Web Application Attack] [Priority: 1] 01/10-10:55:21.438067 63.39.3.100:1208 -> 10.0.0.1:80 TCP TTL:117 TOS:0x0 ID:60940 IpLen:20 DgmLen:576 DF ***A**** Seq: 0xDBAC8A1 Ack: 0x4A65F5A8 Win: 0x2180 TcpLen: 20

Source of the trace:
This trace was taken from my employer's corporate web-hosting DMZ. I am running Snort 1.8.3 with a promiscuous and silent NIC sniffing between the firewall and the Internet router.

Detect was generated by:
Snort Intrusion Detection System. I am running Snort Snarf as a tool to convert the alerts into an html format and then I am running a set of custom shell and PERL scripts to maintain and archive the information. This alert was detected with the following Snort rule:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-ATTACKS
/usr/bin/cc command attempt"; flags:A+; content:"/usr/bin/cc";nocase;
sid:1343; rev:1; classtype:web-application-attack;)
```

Probability the source address was spoofed:
It is very unlikely that this was a spoofed source address as a three-way TCP handshake was completed and the possible attacker issued an HTTP GET.

Description of attack:
This attack uses the Unix cc command in a string contained within the data portion of a

packet.

Attack Mechanism:
This traffic is more along the lines of suspicious activity than an actual attack. The cc command invokes a Unix/Linux C compiler. Invoking this command over the network would indicate a remote session (i.e. telnet) or a deviant packet injected with command strings. Compilers on a compromised server can be very useful to an attacker. An intruder can compile malicious code on your server and use it to further compromise your network or another network. Also an intruder could craft a root kit and modify or replace some of the commonly used binaries on your system. The following excerpt supports the above argument:

“"cc, gcc, perl, python, etc..." Compilers/Interpreter commands
The "cc" and "gcc" commands allow compilation of programs. An attacker may use wget, or tftp to download files, and then use these compilers to compile the exploit. From here anything is possible, including local system exploitation.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/cc%20Phantasmp.c|
Example: http://host/cgi-bin/bad.cgi?doh=gcc%20Phantasmp.c;./a.out%20-p%2031337;

If you see a request for "perl" or "python" it may be possible the attacker downloaded a remote perl or python script, and is trying to locally exploit your system.”

(http://www.cgisecurity.com/papers/fingerprint-port80.txt)

However this particular instance demonstrates a false positive. The string “cc%20” is being searched for by the WEB ATTACKS Snort rule:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-ATTACKS cc
command attempt"; flags:A+; content:"cc%20";nocase; sid:1344; rev:1;
classtype:web-application-attack;)
```

The packet that triggered the alert is displayed below:

```
01/10-10:55:21.438067 63.39.3.100:1208 -> 10.0.0.1:80
TCP TTL:117 TOS:0x0 ID:60940 IpLen:20 DgmLen:576 DF
***A**** Seq: 0xDBAC8A1  Ack: 0x4A65F5A8  Win: 0x2180  TcpLen: 20
47 45 54 20 2F 69 6D 61 67 65 73 2F 6E 72 63 63  GET /images/nrcc
25 32 30 6C 6F 67 6F 2E 67 69 66 20 48 54 54 50  %20logo.gif HTTP
2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 77 77 77 2E  /1.1..Host: www.
67 6F 74 66 2E 6F 72 67 0D 0A 55 73 65 72 2D 41  gotf.org..User-A
67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E  gent: Mozilla/5.
30 20 28 57 69 6E 64 6F 77 73 3B 20 55 3B 20 57  0 (Windows; U; W
69 6E 39 38 3B 20 65 6E 2D 55 53 3B 20 72 76 3A  in98; en-US; rv:
30 2E 39 2E 32 29 20 47 65 63 6B 6F 2F 32 30 30  0.9.2) Gecko/200
31 30 37 32 36 20 4E 65 74 73 63 61 70 65 36 2F  10726 Netscape6/
36 2E 31 0D 0A 41 63 63 65 70 74 3A 20 74 65 78  6.1..Accept: tex
74 2F 78 6D 6C 2C 20 61 70 70 6C 69 63 61 74 69  t/xml, applicati
```

```
6F 6E 2F 78 6D 6C 2C 20 61 70 70 6C 69 63 61 74  on/xml, applicat
69 6F 6E 2F 78 68 74 6D 6C 2B 78 6D 6C 2C 20 74  ion/xhtml+xml, t
65 78 74 2F 68 74 6D 6C 3B 71 3D 30 2E 39 2C 20  ext/html;q=0.9,
69 6D 61 67 65 2F 70 6E 67 2C 20 69 6D 61 67 65  image/png, image
2F 6A 70 65 67 2C 20 69 6D 61 67 65 2F 67 69 66  /jpeg, image/gif
3B 71 3D 30 2E 32 2C 20 74 65 78 74 2F 70 6C 61  ;q=0.2, text/pla
69 6E 3B 71 3D 30 2E 38 2C 20 74 65 78 74 2F 63  in;q=0.8, text/c
73 73 2C 20 2A 2F 2A 3B 71 3D 30 2E 31 0D 0A 41  ss, */*;q=0.1..A
63 63 65 70 74 2D 4C 61 6E 67 75 61 67 65 3A 20  ccept-Language:
65 6E 2D 75 73 0D 0A 41 63 63 65 70 74 2D 45 6E  en-us..Accept-En
63 6F 64 69 6E 67 3A 20 67 7A 69 70 2C 64 65 66  coding: gzip,def
6C 61 74 65 2C 63 6F 6D 70 72 65 73 73 2C 69 64  late,compress,id
65 6E 74 69 74 79 0D 0A 41 63 63 65 70 74 2D 43  entity..Accept-C
68 61 72 73 65 74 3A 20 49 53 4F 2D 38 38 35 39  harset: ISO-8859
2D 31 2C 20 75 74 66 2D 38 3B 71 3D 30 2E 36 36  -1, utf-8;q=0.66
2C 20 2A 3B 71 3D 30 2E 36 36 0D 0A 4B 65 65 70  , *;q=0.66..Keep
2D 41 6C 69 76 65 3A 20 33 30 30 0D 0A 43 6F 6E  -Alive: 300..Con
6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C  nection: keep-al
69 76 65 0D 0A 43 6F 6F 6B 69 65 3A 20 41 53 50  ive..Cookie: ASP
53 45 53 53 49 4F 4E 49 44 47 51 47 51 51 56 56  SESSIONIDGQGQQVV
47 3D 41 42 43 44 4D 49 46 41 4A 49 43 4E 41 4D  G=ABCDMIFAJICNAM
4C 4D 4B 4D 45 41 4C 43 48 4A 0D 0A 52 65 66 65  LMKMEALCHJ..Refe
72 65 72 3A 20 68 74 74                          rer: htt
```

I have highlighted the strings that triggered the alert. After a careful analysis of the data and header portions of this packet I found it to be in every way a normal packet. The communication matched the current technologies that we are using for our web servers and nothing seemed unusual. The content also is a clue to the certainty of the false positive. The link being called is for "nrcc" not "cc", as I knew the content of this particular website I could deduce that this data was innocuous. The threat is also mitigated by the fact that "cc" is a Unix binary and the targeted system was a Windows 2000 Advanced Server.

Correlations:
I have not found any documentation on this specific false positive condition. The only information I could find on the use of "cc%20" was found at: http://www.cgisecurity.com/papers/fingerprint-port80.txt.

Evidence of active targeting:
The remote host definitely targeted our specific hosted website.

Severity:

(4+0)-(5+1) = -2

Criticality: These are my employer's corporate web-hosting servers. Metric equals 4.

Lethality: This is a false positive so I have assigned a metric of 0.

System Countermeasures: We are not running Unix based systems so a metric of 5 is

assigned.

Network Countermeasure: We are taking no network countermeasures at this time. If the attacker wanted to insert strings into a malicious packet we would allow it to pass providing the TCP port was allowed through are firewall.

Defensive recommendation:
Since this is not related to a specific attack, only a general defense can be enumerated. In general these types of code insertion strings are using an exploit such as a buffer overflow or a directory traversal vulnerability. The best defense is well written code and in lieu of that diligent system administration. PERL and CGI have exhibited many of these vulnerabilities; so carefully scrutinizing any deployment of the aforementioned technologies would probably be prudent.

Multiple choice test question:
The Snort rule that caught this string, was searching in:

a) The TCP header to inspect for a malformed packet.
b) The data payload of the TCP packet to find the Windows 2000 "cc" command.
c) The data portion of the layer 4 protocols to find the Unix based "cc" command.
d) The ICMP header to find traces of a Cisco IOS based binary.

Answer: c. This Snort rule is looking for the "cc" string in the data payload portion of the packet and "cc" is a Unix based C compiler.

Detect # 4: ICMP Communication Administratively Prohibited

```
[**] [1:485:2] ICMP Destination Unreachable (Communication
Administratively Prohibited) [**]
[Classification: Misc activity] [Priority: 3]
01/11-10:02:28.988067 65.113.16.22 -> 10.0.0.1
ICMP TTL:239 TOS:0x0 ID:53085 IpLen:20 DgmLen:56
Type:3 Code:13 DESTINATION UNREACHABLE: PACKET FILTERED
** ORIGINAL DATAGRAM DUMP:
10.0.0.1:80 -> 198.26.130.38:16660
TCP TTL:110 TOS:0x0 ID:61730 IpLen:20 DgmLen:52
Seq: 0x53ECFA49
** END OF DUMP
```

```
[**] [1:485:2] ICMP Destination Unreachable (Communication
Administratively Prohibited) [**]
[Classification: Misc activity] [Priority: 3]
01/11-10:02:29.958067 65.113.16.22 -> 10.0.0.1
ICMP TTL:239 TOS:0x0 ID:53096 IpLen:20 DgmLen:56
Type:3 Code:13 DESTINATION UNREACHABLE: PACKET FILTERED
** ORIGINAL DATAGRAM DUMP:
10.0.0.1:80 -> 198.26.130.38:16660
TCP TTL:110 TOS:0x0 ID:61751 IpLen:20 DgmLen:52
Seq: 0x53ECFA49
** END OF DUMP
```

Source of the trace:
This trace was taken from my employer's corporate web-hosting DMZ. I am running Snort 1.8.3 with a promiscuous and silent NIC sniffing between the firewall and the Internet router.

Detect was generated by:
Snort Intrusion Detection System. I am running Snort Snarf as a tool to convert the alerts into an html format and then I am running a set of custom shell and PERL scripts to maintain and archive the information. This alert was detected with the following Snort rule:

Probability the source address was spoofed:
It is very unlikely that this communication involved spoofing. There is evidence of a three–way handshake in the initial communication and further evidence to support the legitimacy of the ICMP message.

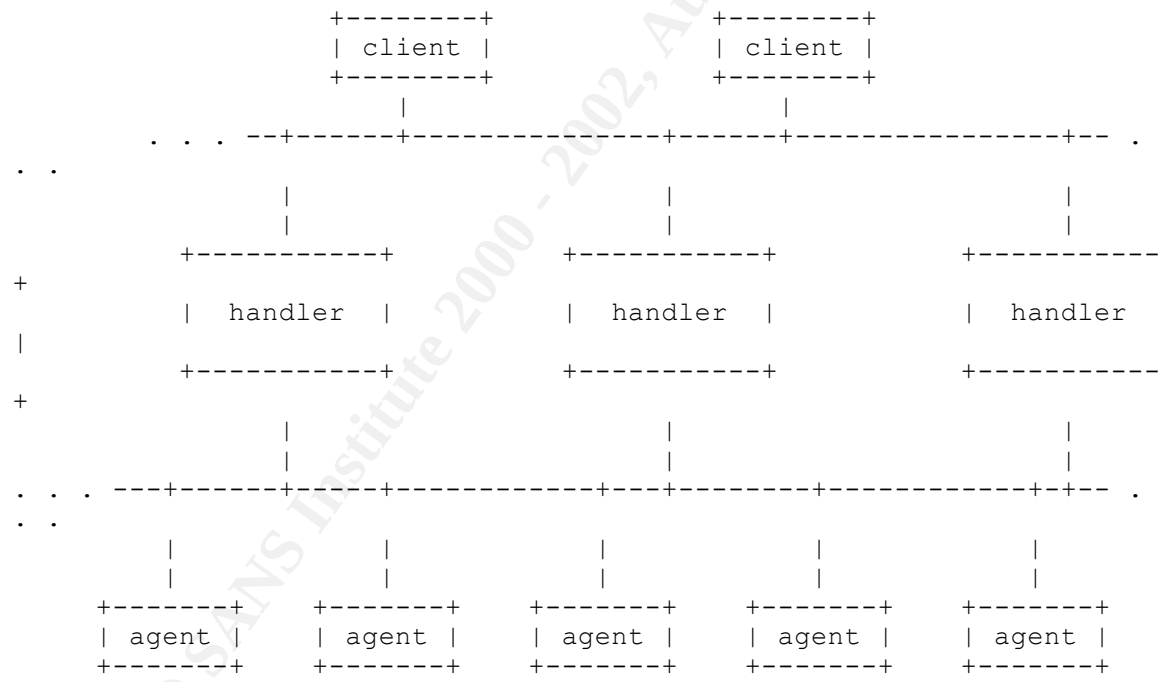
Description of attack:
ICMP Destination Unreachable (Communication Administratively Prohibited) packets are not necessarily evidence of an attack but do indicate that a router or gateway is blocking traffic. (Stevens 1994) (Stacheldraht, which will be discussed in the following paragraphs has been assigned the CVE identification CAN-2000-0138).

Attack Mechanism:
This is an interesting case of false positive. I initially noticed this alert and wondered why a remote network would be blocking traffic coming from my web servers. I began to analyze the alert and found that Snort had copied the headers from the original communication that is sent back inside the ICMP message. Once I had the original communication I looked at the port numbers associated with the conversation. It appeared that my server was communicating on port 80 and that the remote host was on port 16660. At first glance this appeared to be a random port above 1024 that was being used in a standard HTTP conversation with my web server. I also looked at the source of the ICMP message. The source was a different address than the original host and in fact it was a different network. After a series of trace routes it became clear that the ICMP

message came from a border router on the perimeter of the network where the remote host currently resided.

The next question was why did the border router block my web server from responding to the client machine. I began to do a little research into the network that had blocked the packet. It turned out that the network belonged to a branch of the U.S. military (in fact my trace route was knee deep in .mil FQDN's before I even reached the border router in question). Once I knew whom I was dealing with I began to research the TCP port number 16660. It turns out that this TCP port is associated with an attack known as Stacheldraht. Stacheldraht is a Unix based distributed Denial of Service attack tool. The following is an excerpt from the Washington University analysis of Stacheldraht:

"The Stacheldraht network is made up of one or more handler programs ("mserv.c") and a large set of agents ("leaf/td.c"). The attacker uses an encrypting "telnet alike" program to connect to and communicate with the handlers ("telnetc/client.c"). A Stacheldraht network would look like this:



Communication

Client to handler(s): 16660/tcp
Handler to/from agent(s): 65000/tcp, ICMP_ECHOREPLY"

(http://staff.washington.edu/dittrich/misc/stacheldraht.analysis)

It appeared that Stacheldraht infected Unix servers primarily Solaris 2.x and that the nature of the attack was a multi-tiered distributed DOS. As can be seen from the above excerpt port 1660 is used to communicate between the client and the handler. The server in my DMZ did not fit the profile for a Stacheldraht infected server, so I decided to check the firewall logs and see if I could find the initial request from the client to my web server. After some investigation I was able to find the request, it had passed through my firewall and had made it to the web server.

Now I was fairly certain that the server had received a legitimate request from the client although I did not have a packet capture of the HTTP session to analyze. Why then did the router filter my packet? The filtered packet was already part of an established TCP session. The only answer I could deduce was that perhaps I was dealing with non-connection oriented packet filtering. I had become so accustomed to connection oriented packet filtering that the idea had not initially occurred to me. I was not in a position to do any further reconnaissance on the target network so I am not sure what type of filtering is being done. I am not sure if it is a Cisco router, a Unix host acting as a router or a firewall appliance. I could be totally incorrect in my analysis, but without intrusive analysis I could go no further.

Correlations:
Stacheldraht is fairly well documented, the following link contains a detailed discussion: http://staff.washington.edu/dittrich/misc/stacheldraht.analysis. In terms of ICMP Admin. Prohibited there is a detailed discussion in TCP/IP Illustrated Volume I "The Protocols", W. Richard Stevens 1994.

Evidence of active targeting:
Both the HTTP client and the router that sent the ICMP message actively targeted our web server.

Severity:

(4+0)-(5+5) = -6

Criticality: These are my employer's corporate web-hosting servers. Metric equals 4.

Lethality: This is a false positive so I have assigned a metric of 0.

System Countermeasures: Stacheldraht is only a threat to Unix systems. We are running Windows 2000 Advanced Server therefore I have assigned a metric of 5.

Network Countermeasures: If for the sake of argument Stacheldraht compromised one of our servers, communications on TCP port 16660 both ingress and egress would have been blocked by our firewall. Metric equals 5

Defensive recommendation:

In terms of the Stacheldraht attack I can only say that applying basic firewall philosophy of block all traffic by default and then poke holes when justified would help prevent this attack. Most commercially available firewalls are connection oriented; this allows traffic to communicate using ports that might otherwise be blocked. This attack uses other methods of communication including ICMP echo reply. While allowing ICMP echo reply packets into you network from the Internet it can present certain risks such as covert communications (Stacheldraht, Loki etc.). Also making sure that your systems are up to date with all security patches can help to protect your systems from tools such as Stacheldraht.

Multiple choice test question:

Which node initiated the communication?

a) The border router inside the military network started the conversation by sending an ICMP Destination Unreachable (Communication Administratively Prohibited) packet to the web server.
b) The web server initiated the communication by connecting to a client node on TCP port 16660.
c) The client initiated the communication by connecting to the web server with the destination TCP port of 16660 and a source port of 80.
d) The client initiated the communication by connecting to the web server with the destination TCP port of 80 and a source port of 16660.

Detect #5: DNS named version attempt

[**] [1:257:1] DNS named version attempt [**] [Classification: Attempted Information Leak] [Priority: 2] 01/18-00:35:16.858067 217.131.173.105:2454 -> 10.0.0.18:53 UDP TTL:41 TOS:0x0 ID:8147 IpLen:20 DgmLen:58 Len: 38 [Xref => http://www.whitehats.com/info/IDS278]
[**] [1:257:1] DNS named version attempt [**] [Classification: Attempted Information Leak] [Priority: 2] 01/18-00:35:16.868067 217.131.173.105:2454 -> 10.0.0.18:53 UDP TTL:40 TOS:0x0 ID:8147 IpLen:20 DgmLen:58 Len: 38 [Xref => http://www.whitehats.com/info/IDS278]
[**] [1:257:1] DNS named version attempt [**] [Classification: Attempted Information Leak] [Priority: 2] 01/18-00:35:16.868067 217.131.173.105:2454 -> 10.0.0.18:53 UDP TTL:39 TOS:0x0 ID:8147 IpLen:20 DgmLen:58 Len: 38 [Xref => http://www.whitehats.com/info/IDS278]
[**] [1:257:1] DNS named version attempt [**] [Classification: Attempted Information Leak] [Priority: 2] 01/18-00:35:16.868067 217.131.173.105:2454 -> 10.0.0.18:53 UDP TTL:38 TOS:0x0 ID:8147 IpLen:20 DgmLen:58 Len: 38 [Xref => http://www.whitehats.com/info/IDS278]

Source of the trace:
This trace was taken from my employer's corporate web-hosting DMZ. I am running Snort 1.8.3 with a promiscuous and silent NIC sniffing between the firewall and the

Internet router.

Detect was generated by:
Snort Intrusion Detection System. I am running Snort Snarf as a tool to convert the alerts into an html format and then I am running a set of custom shell and PERL scripts to maintain and archive the information. This alert was detected with the following Snort rule:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS named version
attempt"; content:"|07|version"; offset:12; content:"|04|bind";
nocase; offset: 12; reference:arachnids,278; classtype:attempted-
recon; sid:257; rev:1;)
```

Probability the source address was spoofed:
It is very unlikely that this attack involved spoofing. The attacker is trying obtain version information from the server and therefore needs to reveal his true source address.

Description of attack:
This is a reconnaissance attempt that is trying to determine the exact version of BIND that a DNS server is running. The attacker can then apply certain specific attacks such as buffer overflows in an attempt to compromise the server.

Attack Mechanism:
To determine a version of BIND a simple command can be executed:

```
'root@attacker$ dig @dns.victim.com version.bind txt chaos'
```

This command will return the version of BIND running on dns.victim.com.
This particular attacker scanned only one address in our class C subnet range. As we are neither running BIND nor any type of DNS in this subnet, his scan was most likely futile.

[**] [1:257:1] DNS named version attempt [**] [Classification: Attempted Information Leak] [Priority: 2] 01/18-00:35:16.858067 217.131.173.105:2454 -> 10.0.0.18:53 UDP TTL:41 TOS:0x0 ID:8147 IpLen:20 DgmLen:58 Len: 38 [Xref => http://www.whitehats.com/info/IDS278]

After looking at this packet I noticed some indications that this was an automated tool. First I noticed that the intial ID number was consistant with each attempt: 8147. Also the UDP source port remained consistant: 2424. Another curiosity was the TTL, which seemed to decrement with each attempt: 41,40,39,38… I am not sure what tool or script was used to generate this alert.

Correlations:
This reconnaissance and its associated attacks have been well documented. Below are a few good links on the topic:

http://www.computerworld.com/cwi/story/0,1199,NAV47_STO57830,00.html

http://www.cert.org/advisories/CA-2001-02.htmlv

Evidence of active targeting:
The attackers appear to be shooting in the dark. There are no DNS servers residing on that entire class C subnet that we maintain nor are there any servers inadvertently running BIND. However, the attacker picked the 10.10.10.18 address only in his attack and sent around 50 attempts.

Severity:

(4+0)-(5+5) = -6

Criticality: This attack is not diretced at any particular server but it is a vital subnet. Metric equals 4.

Lethality: This is only a probe so is not lethal at this point, metric therfore equals 0.

System Countermeasures: This is a 5, we are not running BIND on our network.

Network Countermeasures: This is a 5, we are blocking all traffic from the Internet on port 53.

Defensive recommendation:
Although we are about as protected from these attacks as you could hope to be, I think a overview of DNS security would be instructional. In general there are several important steps in securing DNS servers. The first step would be to check your version of BIND. If you are not running BIND 8.2.3, or BIND 9.1. it might be a good idea to upgrade. At the very least it would be prudent to check to see what patches are available for your particular version of BIND.

Multiple choice test question:

This probe can be explained by:

a) This is probably just part of a normal DNS communication from inside the firewall to a DNS server on the Internet.
b) The attacker has done initial reconnaissance and knows that a DNS server resides on 10.10.10.18.
c) This attacker is running some kind of tool that is sending version bind packets regardless of whether BIND is running on the target node.
d) The attacker has already compromised the victims DNS server and is communicating on a covert channel.

Answer: c. This is a blind attack, there is not even a live node at the address that is being scanned.

Assignment 3 "Analyze This!" Scenario

Introduction

For this analysis of a University's network I analyzed the files alert.0201[10-14], scans.0201[10-14] and oos_Jan.10.2002.gz. The main approach that I took was varied, in some instances I found that the sheer number of scans or attacks required some investigation. In some instances it was not the amount but the potential severity of the attack. I found that overall this network has quite a high quantity of possible problems. For one it appears that an alarming number of hosts appear to be infected by Internet Worms or Trojans. I have no idea where the IDS was placed on this network, but judging from the fact that so much of the traffic was internal I would guess that either there are multiple NIDS or that the NIDS is placed inside the firewall. It is also more likely that this network has no firewall of any type.

Top 10 alerts:

Rank	Number of Alerts	Alerts	% of total
1	48352	connect to 515 from inside	29.94%
2	30583	ICMP traceroute	18.93%
3	27849	spp_http_decode: IIS Unicode attack detected	17.24%
4	20144	SNMP public access	12.46%
5	8732	Watchlist 000220 IL-ISDNNET-990517	5.40%
6	7115	MISC Large UDP Packet	4.40%
7	6724	INFO MSN IM Chat data	4.16%
8	2836	ICMP Fragment Reassembly Time Exceeded	1.75%
9	2087	High port 65535 udp - possible Red Worm - traffic	1.29%
10	1511	ICMP Router Selection	0.93%

Top 10 talkers:

IP Address	Alerts
MY.NET.150.198	48344 connect to 515 from inside
MY.NET.5.1	30569 ICMP traceroute
MY.NET.153.164	22289 connect to 515 from inside
MY.NET.152.109	5044 SNMP public access
MY.NET.153.185	3870 MISC Large UDP Packet
211.32.117.26	2892 spp_http_decode: IIS Unicode attack detected
MY.NET.153.219	2772 SNMP public access
211.115.213.202	2207 spp_http_decode: IIS Unicode attack detected
207.200.86.66	2006 spp_http_decode: IIS Unicode attack detected
MY.NET.153.112	1870 MISC Large UDP Packet

Analyzed Alerts:

Signature (click for sig info)	# Alerts	# Sources	# Destinations
connect to 515 from inside	48352	64	2
spp_http_decode: IIS Unicode attack detected	27849	83	484
Possible trojan server activity	42	9	9
EXPLOIT x86 NOOP	55	6	5
High port 65535 udp - possible Red Worm - traffic	2087	70	95

Alert Analysis 1

Connect to 515 from inside

Category: Possible False Positive

Analysis:
Port 515 is the listening TCP port used for the Unix LPR or Line Printer. I looked for possible exploits using this port and found that there are several buffer overflow vulnerabilities with Solaris and Linux LPR daemons. The current version of Snort that I am using (1.8.3 with the most recent rule base as of 01-05-02) has specific rules for these vulnerabilities. These rules are shown below:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 515 (msg:"EXPLOIT LPRng
overflow"; flags: A+; content: "|43 07 89 5B 08 8D 4B 08 89 43
0C B0 0B CD 80 31 C0 FE C0 CD 80 E8 94 FF FF FF 2F 62 69 6E 2F
73 68 0A|"; reference:bugtraq,1712; classtype:attempted-admin;
sid:301; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 515 (msg:"EXPLOIT
redhat 7.0 lprd overflow"; flags: A+; content:"|58 58 58 58 25
2E 31 37 32 75 25 33 30 30 24 6E|"; classtype:attempted-admin;
sid:302; rev:1;)
```

It appears that in at least some instances TCP port 721-731 is used as the client end of the LPR communications, as quoted below:

"515 TCP port for LPD. This port can be used when printing with LPD (for example, from UNIX (R)) or using LPD emulation like the Microsoft (R) LPR port monitor. While port **515** is the listen or destination port, TCP ports **721-731** are the source ports on the host machine."
(http://www.aplawrence.com/Jeffl/printports.html)

I found that a variety of ports >1024 were being used but not TCP port 721-731. Another clue was the architecture of the communication: many nodes communicating with only two nodes, MY.NET.150.198 and MY.NET.5.4. This could be a set of network printers receiving print jobs from many nodes on the network. It is possible that these nodes are infected and that they are attempting to infect other servers. This is a case that could be easily resolved by having more information about how the network operated and what operating systems and devices were in deployment.

Sample:

Alert Analysis 2

spp_http_decode: IIS Unicode attack detected

Category: Internet Worm Infection

Analysis:
This looks like a large distributed Internet worm attack. It appears that Snort may be overestimating the actual amount of attacks, as has been indicated by other analysts (http://www.incidents.org/archives/intrusions/msg00797.html). In this case Snort indicates several alarms when there is only one event. In the example below Snort reported 5 events that occurred at the same millisecond:

```
01/10-07:47:29.589358 [**] spp_http_decode: IIS Unicode attack
detected [**] MY.NET.153.111:4520 -> 211.32.117.26:80
01/10-07:47:29.589358 [**] spp_http_decode: IIS Unicode attack
detected [**] MY.NET.153.111:4520 -> 211.32.117.26:80
01/10-07:47:29.589358 [**] spp_http_decode: IIS Unicode attack
detected [**] MY.NET.153.111:4520 -> 211.32.117.26:80
01/10-07:47:29.589358 [**] spp_http_decode: IIS Unicode attack
detected [**] MY.NET.153.111:4520 -> 211.32.117.26:80
01/10-07:47:29.589358 [**] spp_http_decode: IIS Unicode attack
detected [**] MY.NET.153.111:4520 -> 211.32.117.26:80
```

Despite the over representation in the total alerts this appear to be a serious situation. 68 internal addresses appear to be infected and attempting to spread some their malignancy to destinations both internal and out on the Internet. Unicode attacks are found in a number of worms, including Code Red, Code Blue and NIMDA. Without a packet capture it is difficult to tell the exact type of worm. My guess is either NIMDA or Code Red II. NIMDA looks for cmd.exe in /inetpub/wwwroot/scripts, which is basically a server that has already been infected by Code Red.

Correlations:
This alert was also seen by Tom Liston.

```
[**] spp_http_decode: IIS Unicode attack detected [**]
06/13-16:15:52.906181 X.X.X.161:2359 -> 216.214.82.141:80
TCP TTL:115 TOS:0x0 ID:14294 IpLen:20 DgmLen:232 DF
***AP*** Seq: 0xDC8B6DEA  Ack: 0x939B666E  Win: 0x2398  TcpLen: 20
```

Tom had the benefit of a packet capture which identified the attack, in his case it is most likely Code Red.

```
0x0000   4500 00ef 7aef 4000 7306 8a15 425a 9446
E...z.@xxxxxxxxx
0x0010   d8d6 528d 0880 0050 bab1 e085 b1b4 c4bc
..R....P........
0x0020   5018 2398 c120 0000 4745 5420 2f73 6372
P.#.....GET./scr
0x0030   6970 7473 2f2e 2ec0 af2e 2ec0 af2e 2ec0
ipts/...........
0x0040   af2e 2ec0 af2e 2ec0 af2e 2ec0 af2e 2ec0
................
0x0050   af2e 2e61 662e 2e25 6330 2561 662e 2e25
...af..%c0%af..%
```

```
0x0060   6330 2561 662e 2e25 6330 2561 662e 2e25
c0%af..%c0%af..%
0x0070   6330 2561 662f 7769 6e6e 742f 7379 7374
c0%af/winnt/syst
0x0080   656d 3332 2f63 6d64 2e65 7865 3f2f 6325
em32/cmd.exe?/c%
0x0090   3230 7069 6e67 2532 302d 7625 3230 7474        20ping%20-
v%20tt
0x00a0   6c2d 7a65 726f 2d64 7572 696e 672d 7472        l-zero-during-
tr
0x00b0   616e 7369 7425 3230 2d6e 2532 3039 3939        ansit%20-
n%20999
0x00c0   3939 3939 3939 3925 3230 2d6c 2532 3036        9999999%20-
l%206
0x00d0   3535 3030 2532 302d 7725 3230 3025 3230        5500%20-
w%200%20
0x00e0   3231 322e 3530 2e31 3832 2e35 380d 0a
212.50.182.58..
```

(http://www.incidents.org/archives/intrusions/msg00797.html)

Alert Analysis 3

Possible Trojan server activity

Category: Trojan Communication

Analysis:
This communication is a little disturbing; an external host appears to be communicating with an internal host on a port that is assocoiated with the Trojan program Sub Seven. Without a closer look at the packet I am forced to determine the stimulus response sequence by using the time stamp on the Snort capture. Since this is not reliable in terms of which communication occurred first I am given several options. If the external node provided the stimulus then this may be an example of KaZaA (mp3/music files sharing software). In that case the port 27374 may be a random port for a KaZaA communication. Alternatively perhaps an internal box is using a Sub Seven client to control and external machine. It is also possible that I missed a part of this communication and that anexternal box is remotely controlling my internal server.

Sample:

Stimulus:
```
01/12-08:54:35.593668 [**] Possible trojan server activity [**]
80.133.113.3:27374 -> 77.77.150.133:1214
```

Response:
```
01/12-08:54:35.593735 [**] Possible trojan server activity [**]
77.77.150.133:1214 -> 80.133.113.3:27374
```

(http://www.sans.org/y2k/052400-1300.htm)

I was unable to find any scanning activity associated with this communication.

Correlations:
Stephen Northcutt analyzed a similar alert acting as Handler on duty at GIAC.

```
May 20 00:32:30 cc1014244-a kernel: securityalert: tcp if=ef0
  from 24.3.112.18:1797 to 24.3.21.199 on unserved port 27374
```

Alert Analysis 4

EXPLOIT x86 stealth noop

Category: Buffer Overflow

Analysis:
This type of exploit is looking for the noop opcode in the ix86 processor architecture. This technique can be used in buffer overflows. This attack invloves using long strings of noops to overwhelm the processor. This attack can false positive if large binary files are being transferred. If there is is a Trojan injected into the packet it could appear at the end or at any point in the flood of noop code.

In the alert below you can see that the attacker first used a stealth technique and then followed with a series of conspicuous noop traffic. I could find no scans that matched this activity. I am not certain why the alert is marked as stealth, I could not find a Snort rule that matched that message. Perhaps the attacker is usinf SYN/FIN or some other OOS approach to attempt to circumvent a firewall. I could not find any correlations with my OOS captures for the same time period.

Sample:

```
01/10-01:47:32.416491 [**] EXPLOIT x86 stealth noop [**] 24.95.245.166:3982 -
> 77.77.150.190:20
01/10-03:03:21.653983 [**] EXPLOIT x86 NOOP [**] 24.95.245.166:3982 ->
77.77.150.190:20
```

Correlations:
This atack was seen and analyzed by Matt Fearnow acting as Handler on Duty at the Global Incident Analysis Center:

```
Apr  2 21:37:41 hostka snort: EXPLOIT x86 NOOP: 208.227.243.34:3617 -
> a.b.c.225:515
Apr  2 21:37:41 hostka snort: EXPLOIT x86 NOOP: 208.227.243.34:3648 -
> a.b.c.225:515
Apr  2 21:37:45 hostka snort: EXPLOIT x86 NOOP: 208.227.243.34:3819 -
```

```
> a.b.c.225:515
Apr  2 21:37:46 hostka snort: EXPLOIT x86 NOOP: 208.227.243.34:4513 -
> a.b.c.225:515
```

(http://www.sans.org/y2k/040401-1400.htm)

Alert Analysis 5

High port 65535 udp - possible Red Worm - traffic

Category: Possible Worm Infection

Analysis:
The UDP port 65535 is associated with the following Trojans: Adore worm, RC1 trojan, Sins.

There were 8 scans coming from 66.77.13.103 to MY.NET.88.155 between 01/10-11:23:25 and 01/14-16:43:31. I saw 1659 scans to internal destinations associated with the external host 66.77.13.103 using port 0, 7001 and many other high numbered ports:

Jan 14 16:39:21 66.77.13.103:0 -> MY.NET.88.155:0 UDP
Jan 14 16:39:25 66.77.13.103:0 -> MY.NET.88.155:0 UDP
Jan 14 16:39:29 66.77.13.103:0 -> MY.NET.88.155:0 UDP
Jan 14 16:39:34 66.77.13.103:0 -> MY.NET.88.155:0 UDP
Jan 14 16:39:37 66.77.13.103:0 -> MY.NET.88.155:0 UDP

At this point I thought it would be informative to discover the identity of this host:

```
Qwest Cybercenters (NETBLK-QWEST-CYBERCENTER-2)
   1200 Harbor Boulevard
   Weehawken, NJ 07087
   US

   Netname: QWEST-CYBERCENTER-2
   Netblock: 66.77.0.0 - 66.77.127.255
   Maintainer: QCYB

   Coordinator:
      Wysocki, David  (DW820-ARIN)  ip-admin@qis.qwest.net
      201-770-4133

   Domain System inverse mapping provided by:

   DCA-ANS-01.INET.QWEST.NET    205.171.9.242
   SVL-ANS-01.INET.QWEST.NET    205.171.14.195
```

ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

Record last updated on 13-Feb-2001.
Database last updated on 29-Jan-2002 19:56:39 EDT.

I traced this host back to the QWEST-CYBERCENTER. QWEST from my experience is a large telecommunciations company that provides private lines such T1 and T3 circuits to large communications. The QWEST Cybercenter appeared to be a large web-hosting division of QWEST. I attempted to do a reverse lookup on the host but received:

Error: Host not found.
rcode = 3 (Non-existent domain), ancount=0

One anomaly with this traffic is the fact that the source uses different ports to connect in a fairly small time period. As you can see in the sample (below) the first UDP port is 32767 and the next one 11 seconds later is 65535. I know that many trojans work on ports higher than 32767, but I could not isolate this particular Trojan in my research. Although is would appear that at least 24 internal hosts are infected I did not find any evidence of a two way communication between an internal host and external host on the Internet on these high numbered UDP ports. What I did see is a large amount of traffic from internal servers to other internal servers. The nodes I saw communicating on UDP port 65535 were either communicating with or trying to infect other internal nodes.

Sample:

```
01/14-12:11:38.064486 [**] High port 65535 udp - possible Red
Worm - traffic [**] 66.77.13.103:32767 -> 77.77.88.155:65535

01/14-12:11:49.953210 [**] High port 65535 udp - possible Red
Worm - traffic [**] 66.77.13.103:65535 -> 77.77.88.155:65384
```

Correlations:
This type of activity was seen by Jens Hector:

> Feb 29 07:12:25 firepower kernel: Packet log: private1
> DENY eth0 PROTO=6
> 192.115.221.125:65535 207.245.232.127:65535 L=28 S=0x00
> I=15817 F=0x00B8 T=47

(http://cert.uni-stuttgart.de/archive/incidents/2000/04/msg00013.html)

Scans

Top 10 Destinations of Scans:

Number of Scans	IP Address
34214	MY.NET.1.3
27415	MY.NET.88.155
26191	MY.NET.1.4
15089	MY.NET.6.45
14995	MY.NET.60.43
14926	MY.NET.153.173
13588	MY.NET.153.143
13494	MY.NET.153.204
13396	MY.NET.153.163
13270	MY.NET.151.17

Top 10 Sources for scans:

Number of Scans	IP Address
395749	MY.NET.60.43
51104	MY.NET.6.49
42460	MY.NET.6.52
38674	MY.NET.150.143
32096	MY.NET.6.50
31558	MY.NET.6.45
27388	MY.NET.6.48
24928	205.188.228.65
20690	MY.NET.6.51
19408	205.188.228.17

Scan analysis 1

Category: External Scan

Analysis:

One alarming indication was that an external address showed up in the top ten scans: 205.188.228.65. It was calculated to represent 24,928 instances. I did a trace of this address and I was not surprised when I saw the source:

```
America Online, Inc (NETBLK-AOL-DTC)
22080 Pacific Blvd
Sterling, VA 20166
US

Netname: AOL-DTC
```

Netblock: 205.188.0.0 - 205.188.255.255

```
   Coordinator:
      America Online, Inc.   (AOL-NOC-ARIN)   domains@AOL.NET
      703-265-4670

   Domain System inverse mapping provided by:

   DNS-01.NS.AOL.COM        152.163.159.232
   DNS-02.NS.AOL.COM        205.188.157.232

   Record last updated on 27-Apr-1998.
   Database last updated on  29-Jan-2002 19:56:39 EDT.
```

I also did a reverse lookup and found the results:

65.228.188.205.in-addr.arpa name = mslb4.spinner.com.
I then did a lookup of spinner.com:

```
Domain Name: SPINNER.COM

Registrant:
  Spinner Networks, Inc.
    1209 Howard Ave Suite 200
    Burlingame, CA 90410
    US

  Created on..............: Dec 23, 1999
  Expires on..............: Dec 23, 2001
  Record Last Updated on..: Jan 05, 2000
  Registrar...............: America Online, Inc.
  http://whois.registrar.aol.com/whois/

  Administrative Contact:
    Domain Administration, Spinner
    Spinner Networks, Inc.
    1209 Howard Ave Suite 200
    Burlingame, CA 90410
    US
    Email. hostmaster@SPINNER.COM
    Tel. 415 934 2700
    Fax. 415 934 2756
  Technical Contact:
    Domain Administration, Spinner
    Spinner Networks, Inc.
    1209 Howard Ave Suite 200
    Burlingame, CA 90410
    US
    Email. hostmaster@SPINNER.COM
    Tel. 415 934 2700
    Fax. 415 934 2756
  Domain servers:
```

```
dns-01.spinner.net
  152.163.159.239
dns-02.spinner.net
  205.188.157.239
```

It turns out that this is a streaming audio service owned by AOL. A close inspection of the port numbers it appeared that all the destinations were on TCP port 6679 or Real Time Protocol. The reason Snort caught this, as a scan is no doubt due to the bandwidth and synchronous communications required for streaming media. If I were in charge of this network I might consider stopping this type of traffic, however as this is a University that might not be a practical solution.

Sample:

Jan 10 08:37:13 205.188.228.65:24824 -> MY.NET.151.85:6970 UDP
Jan 10 08:37:13 205.188.228.65:15740 -> MY.NET.151.105:6970 UDP
Jan 10 08:37:13 205.188.228.65:31376 -> MY.NET.151.17:6970 UDP
Jan 10 08:37:13 205.188.228.65:24958 -> MY.NET.151.79:6970 UDP

Scan analysis 2

Category: Possible Compromise

Analysis:
There are 130,217 involving UDP ports 7000 and 7001. This traffic is mostly internal but there are several examples of Internet traffic coming inside the MY.NET network. An example of this can be seen below:

```
Jan 10 11:37:00 66.77.13.117:7001 -> MY.NET.88.155:7000 UDP
Jan 10 11:37:03 66.77.13.117:7000 -> MY.NET.88.155:7001 UDP
Jan 10 11:37:06 66.77.13.117:7000 -> MY.NET.88.155:7001 UDP
Jan 10 11:37:19 66.77.13.117:7000 -> MY.NET.88.155:7001 UDP
Jan 10 11:37:22 66.77.13.117:7000 -> MY.NET.88.155:7001 UDP
```

There are several reason for this type of traffic, one could be a scan by a tool such as Remote Grabber. Remote Grabber allows you to view the screen of a user remotely. This is used for surveillance, but is also a good way to grab password and credit card numbers. It is also possible that these ports are being used by AFS. AFS is a distributed file system that can operate in a LAN or WAN environment. (http://www.faqs.org/faqs/afs-faq/). The following ports apply to AFS:

afs3-fileserver 7000/tcp file server itself
afs3-fileserver 7000/udp file server itself
afs3-callback 7001/tcp callbacks to cache managers
afs3-callback 7001/udp callbacks to cache managers

The fact that this traffic is so numerous indicates to me that this is probably a deployment

of AFS. Some type of extranet configuration could explain the fact that an Internet address is also seen. I did an ARIN lookup on the Internet host and it came back as a QWEST address block. This is the second time I have found a host inside QWEST's network. It is possible that they are this institution's ISP.

```
Qwest Cybercenters (NETBLK-QWEST-CYBERCENTER-2)
   1200 Harbor Boulevard
   Weehawken, NJ 07087
   US

   Netname: QWEST-CYBERCENTER-2
   Netblock: 66.77.0.0 - 66.77.127.255
   Maintainer: QCYB

   Coordinator:
      Wysocki, David  (DW820-ARIN)  ip-admin@qis.qwest.net
      201-770-4133

   Domain System inverse mapping provided by:

   DCA-ANS-01.INET.QWEST.NET    205.171.9.242
   SVL-ANS-01.INET.QWEST.NET    205.171.14.195

   ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

   Record last updated on 13-Feb-2001.
   Database last updated on  29-Jan-2002 19:56:39 EDT.
```

Below you can see excerpts from two way communications between first internal and then external to internal hosts:

```
Jan 10 10:35:12 MY.NET.153.160:7001 -> MY.NET.6.52:7000 UDP
Jan 10 10:35:12 MY.NET.6.52:7000 -> MY.NET.152.178:7001 UDP

Jan 14 12:21:10 66.77.13.103:7000 -> MY.NET.88.155:7001 UDP
Jan 14 12:21:09 66.77.13.103:7001 -> MY.NET.88.155:7000 UDP
Jan 14 12:21:14 66.77.13.103:7000 -> MY.NET.88.155:7001 UDP
```

Scan analysis 3

Null scan!

Category: Reconnaissance

Analysis:
This apears to be a connection attempt and scan from the Internet to an internal server. These port numbers are associated with KaZaA and are probably attempts to gain access to a host that has installed KaZaA file sharing software. I am not sure why KazaA would

be using a NULL scan. It is likely that a person is scanning for a specific response or open port. I have seen no instances of response to this stimuli. Below is a current Snort rule that covers NULL scans.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN
NULL";flags:0; seq:0; ack:0; reference:arachnids,4;
classtype:attempted-recon; sid:623; rev:1;)
```

Sample:

```
Jan 12 16:32:55 24.112.229.150:2943 -> MY.NET.150.145:1214 SYN
******S*
Jan 12 16:32:58 24.112.229.150:2943 -> MY.NET.150.145:1214 NULL
********
Jan 12 16:53:23 24.112.229.150:3768 -> MY.NET.150.145:1214 NULL
********

Jan 10 17:56:23 24.200.175.145:3587 -> MY.NET.150.133:1214 NULL
********
Jan 10 18:10:24 24.200.175.145:70 -> MY.NET.150.133:4240 NULL
********
```

I could not find any trace of these NULL scans in the OOS logs for the same time period. I took the time to gather information on the scanner and she appears to be coming from inside a broadband cable provider.

Rogers@Home Pr (NETBLK-ON-ROG-PRDG-6)
1 Mount Pleasant Road
Toronto, ON M4Y 2Y5
CA

Netname: ON-ROG-PRDG-6
Netblock: 24.112.228.0 - 24.112.231.255

Coordinator:
Budd, Paul (AD30-ARIN) abuse@rogers.com
(416) 935-4729

Record last updated on 18-Mar-1999.
Database last updated on 29-Jan-2002 19:56:39 EDT.

Scan analysis 4

MY.NET.60.43:123 -> MY.NET.x.x

Category: Possible False Alarm

Analysis:

This is possibly just a time server responding to time sync requests. UDP port 123 is used by the network time protocol (NTP). However there is an attack named Net Controller that also uses port 123 (http://www.simovits.com/trojans/tr_data/y1142.html). However that attack is associated with TCP port 123 not UDP port 123. The only problem with the hypothesis that MY.NET.60.43 is an NTP server is the fact that all I see is stimulus and no response. I know that some NTP configurations involve broadcasts to a given network, however these are multicast packets. The only way this could be a broadcast from an NTP server would be if a router on this network was using IP Directed Broadcasts. However if that were the case I would expect to see the receiving destinations listening on the same port, not a series of ports; in fact, the same could be said for any NTP communication.

Sample:

```
Jan 10 00:26:23 MY.NET.60.43:123 -> MY.NET.153.201:3485 UDP
Jan 10 00:26:23 MY.NET.60.43:123 -> MY.NET.153.191:1267 UDP
Jan 10 00:26:23 MY.NET.60.43:123 -> MY.NET.153.151:1691 UDP
Jan 10 00:26:25 MY.NET.60.43:123 -> MY.NET.88.148:1378 UDP
```

Out of Spec:

SYN-FIN scan!

Category: SYN Scan (OOS)

Analysis:

I found a number of OOS alerts:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+
01/10-12:47:28.952386 130.161.249.59:22 -> MY.NET.153.109:22
TCP TTL:20 TOS:0x0 ID:39426
**SF**** Seq: 0x1AE2E35E   Ack: 0x2506636C   Win: 0x404
00 00 00 00 00 00                                  ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+
01/10-12:47:28.972990 130.161.249.59:22 -> MY.NET.153.110:22
TCP TTL:27 TOS:0x0 ID:39426
**SF**** Seq: 0x1AE2E35E   Ack: 0x2506636C   Win: 0x404
00 00 00 00 00 00                                  ......
```

Each of these attacks has the SYN/FIN flags set and have a source and destination port of

TCP 22. SYN/FIN scans allegedly can penetrate certain firewalls and evade some logging systems. (Northcutt and Novak, 1994 p. 226). I have found a piece of code attributed to "jackal" that claims it can subvert some firewall using SYN/FIN scans. These type of malformed packets can also be used for operating system fingerprinting tools such as NMAP and Queso (http://www.insecure.org/).

This appears to be a fast scan looking for vulnerable SSH daemons. This can be a serious type of attack. It uses a synscan technique that can be part of an attack like ramen. The ramen worm could attack a vulnerable host as soon as the scanning binary found a vulnerable host and the scanner could continue to operate.
(http://lists.jammed.com/incidents/2001/10/0104.html)

Correlation:
Can Erkin Acar saw the same type of traffic into his network:

>Oct 16 19:54:25.228427 XXX.XXX.XXX.XXX.22 > YYY.YYY.YYY.YYY.22: SF [tcp sum ok] 415795998:415795998(0) win 1028 (ttl 27, id 39426)

> Oct 16 19:54:26.573878 XXX.XXX.XXX.XXX.1845 > YYY.YYY.YYY.YYY.22: S [tcp sum ok] 4137188806:4137188806(0) win 32120 <mss 1460,sackOK,timestamp 164825588 0,nop,wscale 0> (DF) (ttl 49, id 30236)

(http://lists.jammed.com/incidents/2001/10/0104.html)

The only discrepancy I have found with Can's account is the follow-up scan that actually attempts to connect or at least grab banner information. I have not found any evidence of the follow-up packet at this time. I was not able to determine what tool is being used but I would check to see if any of the scanned boxes are running SSH services. If so, then SSH should be checked for version updates, patches and new vulnerabilities.

Sample:

```
Jan 10 12:47:11 130.161.249.59:22 -> MY.NET.151.14:22 SYNFIN
******SF
Jan 10 12:47:11 130.161.249.59:22 -> MY.NET.151.17:22 SYNFIN
******SF
Jan 10 12:47:11 130.161.249.59:22 -> MY.NET.151.18:22 SYNFIN
******SF
Jan 10 12:47:11 130.161.249.59:22 -> MY.NET.151.21:22 SYNFIN
******SF
```

I also looked to see who was assigned these addresses:

```
Technische Universiteit Delft (NET-DUT-LAN)
   Dienst Technische Ondersteuning
   2600 AJ Delft,
   NL
```

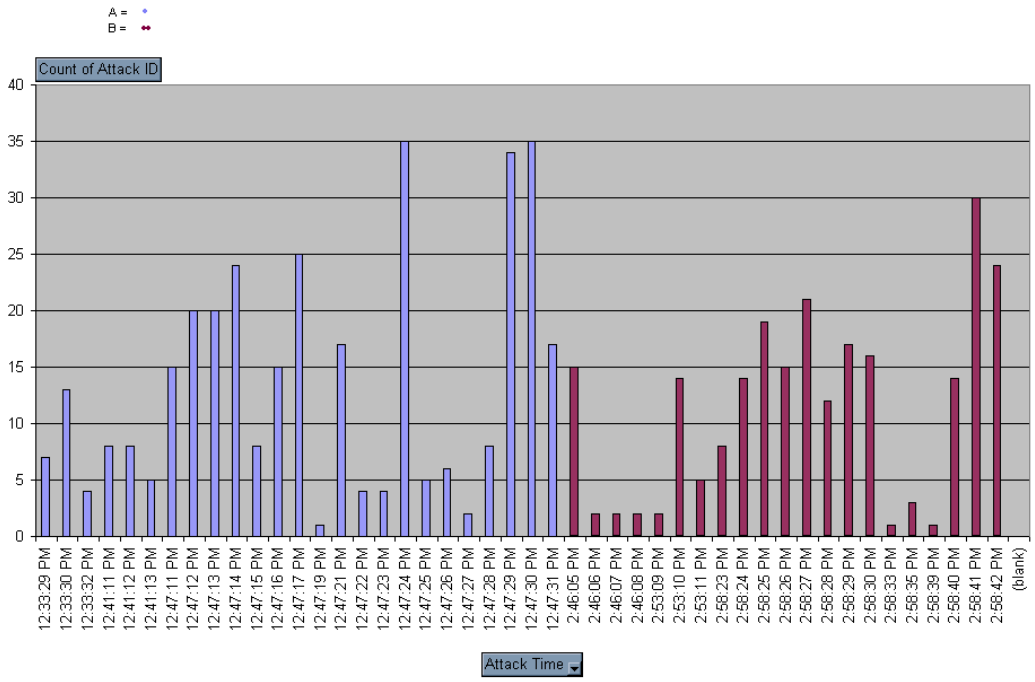
```
Netname: DUNET
Netblock: 130.161.0.0 - 130.161.255.255

Coordinator:
   Kruijf, Freek de   (FD18-ARIN)  SSC@TUDelft.nl
   +31 15 2783226 (FAX) +31 15 2783787

Domain System inverse mapping provided by:

NS1.TUDELFT.NL          130.161.180.1
NS2.TUDELFT.NL          130.161.180.65
NS1.SURFNET.NL          192.87.106.101
NS1.ET.TUDELFT.NL       130.161.33.17

Record last updated on 10-Nov-2000.
Database last updated on  29-Jan-2002 19:56:39 EDT.
```

This traffic is more suspicious to due to the fact that this address range is part of a university and it is from the Netherlands. I also made a grah detailing the time spread of these attacks. The attacks were attempted by two sources, the one seen above and 195.42.223.134. That attackers information is shown below:

```
inetnum:      195.42.223.128 - 195.42.223.159
netname:       WINEASY-EASYNET-MWEB
descr:         mWeb
country:       SE
admin-c:       PA2583-RIPE
tech-c:        PA2583-RIPE
rev-srv:       ns.wineasy.se
rev-srv:       ns2.wineasy.se
status:        ASSIGNED PA
notify:        t.bjorklund@wineasy.se
mnt-by:        RIPE-NCC-NONE-MNT
changed:       m.taskinen@wineasy.se 20000315
source:        RIPE
route:        195.42.192.0/19
descr:         WINEASY
origin:        AS12352
notify:        t.bjorklund@wineasy.se
mnt-by:        AS12352-MNT
changed:       t.bjorklund@wineasy.se 19990618
source:        RIPE
person:       Pontus Axelsson
address:       Drottninggatan 110
address:       113 60 Stockholm
address:       Sweden
phone:         +46 8 54542323
fax-no:        +46 8 54542322
nic-hdl:       PA2583-RIPE
changed:       m.taskinen@wineasy.se 20000315
source:        RIPE
```

The graph shows the distribution of attempts over time. The two attackers have been abbreviated: A = 130.161.249.59 and B = 195.42.223.134. "A" attacked on 01-10-02 and

"B" attacked on 01-14-02.



In this case I was looking for signs of an automated attack. Attacker "A" scanned for 14 minutes and attacker "B" scanned for 12 minutes. In addition to length of scan I looked for a pattern or sequence in the time spread of the scans. Unfortunately, I would need more data to really profile this particluar attack sequence. In terms of destination IP groupings the scans seemed to focus on large blocks within subnets. Either the capture is missing data or the groupings are not consistant. The blocks ranged from 45 to 164 following a class C range. For example MY.NET.5.x and MY.NET.150.x are some of the class C ranges that the tool targeted.

Defensive Recommendations:

Overall this network is in need of an intense security audit. The first step would be to install and configure a firewall between the internal hosts and the Internet. The majority of the problems I see on this network can be traced to a lack of IP packet filtering. Many of these boxes are infected with what appears to be Code Red II | NIMDA and several other possible worms or Trojans. These servers are probing nodes outside the internal network in an attempt to infect others. As a public policy measure, shutting of this network to the world on specific ports even for a short period of time may be required. Second, an enterprise wide audit of devices and operating sytems need to done. It would appear that an inordinate number of boxes are responding on port 80; this could probably be changed. All computers on the network need to audited for operating system and application patch levels. Also drafting policy regarding the use of chat clients and multimedia software might not be a bad idea.

Some other steps that could be taken include access lists and packet filtering at the boundary and the internal routers. Many Worm attacks can be filtered at the router before even making it to a firewall, whethet the traffic is internal or external (http://www.cisco.com/warp/public/63/nimda.shtml). My general observations of this network are congruent with what I have seen at large education institutions: no firewalls or packet filtering, poorly administrated servers and many legacy systems running in paralell with unpatched products from certain prominent software manufacturers. At the very least auditing all nodes for software vulnerabilities and current compromises is highly reccomended.

Analysis process:
In terms of analysis I relied on a collection of commands and utilities.

Alerts:
For the alerts I first concatenated the files using the Unix 'cat' command:

' cat *.alert > alerts'

Then I ran the PERL utility SnortSnarf against the new alerts file:

' snortsnarf -d /reports/directory alerts'

I used these reports to enumerate and analyze the alerts. SnortSnarf can be downloaded for free at: http://www.silicondefense.com/software/snortsnarf/index.htm.

Scans:
Although theoretically SnortSnarf can be used I found that the file sizes once I concatenated were too big for my little Linux workstation. Instead I used a set of scripts I found in the GCIA practical submitted by Gregory Lajon:

"Trimming the scan logs

```
grep -e'^Sep' scans |awk '{print $4" "$6" "$7" "$8}' > allscans
```

sorting top source scans

```
awk '{print $1}' allscans | awk -F':' '{print $1}' |sort | uniq -c |
sort -nr |head -10> src_scans
```

sorting top 10 destination port scans

```
awk '{print $2}' allscans | awk -F':' '{print $1}' |sort | uniq -c |
sort -nr|head -10 > dst_scans
```

sorting top dest port in scans :

```
awk '{print $2}' allscans | awk -F':' '{print $2}' |sort | uniq -c |
sort -nr > dstport_scans
```

List of alerts for which an IP is involved. src_i contains the list of IPs to check

```
for i in `cat src_i`; do echo $i >>list.txt ; grep $i alerts | sed -e
's/\[\*\*\]/\*\*/g' |awk -F'**' '{print $2}' | sort | uniq -c | sort -
nr >> list.txt; done"
```

(Gregory Lajon, GIAC Intrusion Detection In Depth, 2001 Practical assignment Version 3.0)

I then used the following commands to pick out specific IP's and ports:

'cat scans |grep MY.NET.60.43 > MY.NET.60.43'

I used the same techniques for the OOS logs. Then I used Microsoft Excel to create graphs and manipulate large blocks of data.

References:

[1] http://www.securiteam.com/securitynews/A_new_stealth_port_scanning_method.html

[2] http://rr.sans.org/intrusion/spoof.php

[3] http://rr.sans.org/audit/port_scan.php

[4] http://the.wiretapped.net/security/network-mapping/gps/gps-README.txt

[5] http://gps.sourceforge.net/

[6] http://rr.sans.org/audit/hping2.php

[7] http://www.cert.org/advisories/CA-2001-09.html

[8] http://www.securiteam.com/tools/3G5PWR5QAM.html

[9] W. Richard Stevens, 1994. *TCP/IP Illustrated, Volume 1 The Protocols*. New York Addison-Wesley

[10] Stephen Northcutt and Judy Novak , 2000. *Network Intrusion Detection An Analysts Handbook*. Indianapolis: New Riders

[11] http://www.cert.org/incident_notes/IN-2001-09.html

[12] http://www.cert.org/advisories/CA-2001-13.html

[13] http://www.cert.org/incident_notes/IN-2001-09.html

[14] http://www.incidents.org/react/code_redII.php

[15] http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms01-033.asp

[16]http://www.kb.cert.org/vuls/id/111677

[17] http://www.silicondefense.com/software/snortsnarf/index.htm

[18] http://www.incidents.org/archives/intrusions/msg00797.html

[19] http://www.cisco.com/warp/public/63/nimda.shtml

[20] Gregory Lajon, GIAC Intrusion Detection In Depth, 2001 Practical assignment Version 3.0

[21] http://www.icir.org/vern/papers/norm-usenix-sec-01-html/node8.html

[22] http://www.cert.org/advisories/CA-2001-15.html

[23] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0353

[24] http://www.sans.org/y2k/111000-1200.htm

[25] http://www.sans.org/y2k/061800-1900.htm

[26] http://www.giac.org/practical/David_Oborn_GCIA.html

[27] http://www.simovits.com/trojans/tr_data/y1401.html

Appendix I:

Total Snort Alerts parsed by SnortSnarf:

Signature (click for sig info)	# Alerts	# Sources	# Destinations	
WEB-IIS iisadmpwd attempt	1	1	1	
ICMP SRC and DST outside network	1	1	1	
MISC PCAnywhere Startup	1	1	1	

Incomplete Packet Fragments Discarded	1	1	1	
Probable NMAP fingerprint attempt	1	1	1	
FTP MKD - possible warez site	1	1	1	
WEB-CGI bb-hist.sh access	1	1	1	
WEB-IIS 5 .printer isapi	1	1	1	
MISC Large ICMP Packet	1	1	1	
INFO Outbound GNUTella Connect accept	1	1	1	
TFTP - Internal UDP connection to external tftp server	1	1	1	
00:17:00.845119	1	1	1	
WEB-IIS Unauthorized IP Access Attempt	2	1	1	
Tiny Fragments - Possible Hostile Activity	2	1	1	
WEB-IIS admin access	2	1	1	
FTP CWD - possible warez site	2	2	1	
EXPLOIT x86 stealth noop	4	3	3	
TFTP - External UDP connection to internal tftp server	4	2	1	
Port 55850 tcp - Possible myserver activity - ref. 010313-1	4	3	3	
ICMP Echo Request Cisco Type.x	5	1	1	
Attempted Sun RPC high port access	5	5	5	
Back Orifice	6	4	6	
FTP MKD / - possible warez site	7	1	1	
RPC tcp traffic contains bin_sh	8	2	2	
SCAN FIN	9	4	3	
EXPLOIT x86 setgid 0	11	10	9	
INFO - Possible Squid Scan	13	2	3	
EXPLOIT x86 setuid 0	14	6	4	
WEB-FRONTPAGE _vti_rpc access	15	4	1	
INFO Inbound GNUTella Connect accept	15	1	15	
EXPLOIT NTPDX buffer overflow	15	8	3	
SCAN Synscan Portscan ID 19104	16	16	4	
ICMP Echo Request CyberKit 2.2 Windows	17	6	7	
Watchlist 000222 NET-NCFC	18	2	2	
WEB-MISC 403 Forbidden	20	1	6	
WEB-COLDFUSION administrator access	20	1	1	
INFO Inbound GNUTella Connect request	23	11	1	
WEB-IIS _vti_inf access	24	7	2	
INFO Napster Client Data	26	5	6	
INFO Possible IRC Access	29	7	6	

ICMP Destination Unreachable (Protocol Unreachable)	30	2	2	
WEB-MISC compaq nsight directory traversal	32	7	7	
NMAP TCP ping!	34	6	2	
ICMP Destination Unreachable (Host Unreachable)	35	1	21	
IDS552/web-iis_IIS ISAPI Overflow ida nosize [arachNIDS]	41	36	7	
Possible trojan server activity	42	9	9	
EXPLOIT x86 NOOP	55	6	5	
ICMP Echo Request Nmap or HPING2	60	5	2	
spp_http_decode: CGI Null Byte attack detected	60	3	8	
TCP SRC and DST outside network	80	11	8	
SCAN Proxy attempt	97	24	9	
ICMP Echo Request BSDtype	101	1	1	
WEB-IIS view source via translate header	102	2	1	
INFO - ICQ Access	148	2	16	
INFO FTP anonymous FTP	171	21	21	
MISC traceroute	190	4	3	
WEB-MISC Attempt to execute cmd	217	16	6	
FTP DoS ftpd globbing	266	6	5	
ICMP Destination Unreachable (Communication Administratively Prohibited)	299	1	1	
Null scan!	331	61	7	
ICMP Echo Request Windows	473	13	18	
SYN-FIN scan!	601	3	384	
ICMP Echo Request L3retriever Ping	658	23	13	
SMB Name Wildcard	1160	48	46	
ICMP Router Selection	1511	134	1	
High port 65535 udp - possible Red Worm - traffic	2087	70	95	
ICMP Fragment Reassembly Time Exceeded	2836	19	45	
INFO MSN IM Chat data	6724	58	58	
MISC Large UDP Packet	7115	8	6	
Watchlist 000220 IL-ISDNNET-990517	8732	15	5	
SNMP public access	20144	16	138	
spp_http_decode: IIS Unicode attack detected	27849	83	484	
ICMP traceroute	30583	6	3	
connect to 515 from inside	48352	64	2	