GIAC Intrusion Analyst Certification Practical
For Mark Fuster

Assignment 1

Detect 1

(Tom Vandepoel)
Service scan for a port I've never seen before. Anyone have an idea what this is?

```
Aug 30 01:52:51 207.233.243.100:3384 -> my.class.c.42:635 SYN **S*****
Aug 30 01:52:51 207.233.243.100:3483 -> my.class.c.141:635 SYN **S*****
Aug 30 01:52:51 207.233.243.100:3526 -> my.class.c.184:635 SYN **S*****
Aug 30 01:52:51 207.233.243.100:3426 -> my.class.c.84:635 SYN **S*****
Aug 30 01:52:52 207.233.243.100:3550 -> my.class.c.208:635 SYN **S*****
Aug 30 01:52:52 207.233.243.100:3551 -> my.class.c.209:635 SYN **S*****
Aug 30 01:52:52 207.233.243.100:3552 -> my.class.c.210:635 SYN **S*****
Aug 30 01:52:52 207.233.243.100:3553 -> my.class.c.211:635 SYN **S*****
Aug 30 01:52:52 207.233.243.100:3634 -> my.class.c.37:635 SYN **S*****
Aug 30 01:52:52 207.233.243.100:3625 -> my.class.c.28:635 SYN **S*****
```

Network Access Solutions (NETBLK-DIGINET333)
100 Carpenter Drive Suite 206
Sterling, VA 20164 US
Netname: DIGINET333
Netblock: 207.233.243.96 - 207.233.243.103
Coordinator:
Network Operations Center (PCI-ORG-ARIN)
703-736-9800 Fax- 703-736-1654
Record last updated on 22-Nov-1999.
Database last updated on 30-Aug-2000 17:58:01 EDT.

1. Source of Trace

http://www.sans.org/y2k/090100.htm

2. Detect was generated by:

```
Aug 30 01:52:51 207.233.243.100:3384 -> my.class.c.42:635 SYN **S*****
```

This trace appears to come from Snort. The format separated by space delimiters is:

Timestamp (assume local time); source IP and port; destination IP and port; TCP Flags

3. Probability the source address was spoofed:

In my opinion, it is unlikely that the source IP is spoofed. This scan could be used to determine the existence of a host or the existence of a service. Either way, the source host would need to receive the information.

4. Description of Attack:

This packet trace shows a TCP connection request as defined by the initial step in a 3-way TCP handshake. This is not in of itself unusual though an attempt to mount a file system (service defined by port 653) from an external IP is suspicious. This network activity is probably a reconnaissance type activity. A potential hacker could use this port scan for several purposes. First, it could be used for network mapping. By scanning the entire class C address space, it could be used to determine the existence of a host. Ordinarily, if a host not blocked by a perimeter device would respond with a TCP RESET (TCP packet with RST and ACK flags set on both Linux (Mandrake 7.1) or Windows 95 OSR2 TCP/IP stacks) if the service was not open, as witnessed by the following sample tcpdump trace taken from a Mandrake 7.1 host "b."

```
09:24:26.271852 a.1028 > b.635: S 841416:841416(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]>
(DF) [tos 0x7]
09:24:26.271956 b.635 > a.1028: R 0:0(0) ack 841417 win 0 [tos 0x7]
```

If the service was open on the target host, the 2nd part of the 3-way handshake would occur. In this case, since neither was received, it might tell a potential hacker that the hosts are behind a packet filter or firewall. Since the log is not descriptive, it is entirely possible, however, that the log is actually from a perimeter device that blocked the TCP connection request and did not reply back to the source host.

A second possible purpose is to determine whether the target host is running a "Unix-like" system. The mount daemon is used by systems supporting NFS/NIS (Network File System/Network Information System) and is available only on "Unix-like" machines. If the target replies with a SYN/ACK, then the potential hacker has learned something. If the potential hacker is really lucky, an improperly configured host may allow his filesystems to be mounted by any host (e.g. -p or promiscuous mode that accepts requests from any host). Once the potential hacker has learned that the target is a "Unix" box, they may also go after it with other exploits, such as rpc exploits.

5. Attack Mechanism:

This packet trace shows a TCP connection request as defined by the initial step in a 3-way TCP handshake. This network activity is probably a reconnaissance type activity. A potential hacker could use this initial request to network or port map a range of addresses. It could also be used to help fingerprint a host (e.g. this is a "Unix-like" host). If the potential hacker is really lucky, they might gain access to a file system, if the target is exporting file systems, and modify it (i.e. delete files, upload a different configuration file /etc/exports, a rootkit for future use, etc.).

It would appear that the attacker is using an automated script given the number of requests in the period of time. The source port numbers are also grouped and in some cases are sequential. All port numbers differ from the previous by at most two digits in increments of one (e.g. source ports 3550-3553 scanning target ports 208-211). The time of the attack is also outside of "normal" business hours.

6. Correlations:

This detect was attributed to (Tom Vandepoel)

7. Evidence of Active Targeting:

From the number of scans, it would appear that the class C address space of the potential target site is being targeted. It is possible that the potential hacker plans to return later to map the rest of the space slowly. Other colleagues may assist him and he could return with a different type of port scan.

8. Severity:

Typically, the following formula is used to calculate the severity of the attack. Points are assigned on a five point scale, 5 being the highest.

Severity = (Target criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures)

Applying the formula to exploits taken from the SANS site is a little problematic as it is not at all clear, if the author's site has any system or network countermeasures. In some cases, the target criticality is also not easily discernable because of a lack of familiarity again with the particular site's configuration.

Target Criticality: In this case, targets did not seem to be targeted specifically, so assign Target criticality = 2. Attack Lethality: While there is nothing lethal in this reconnaissance as shown in the trace, should the attacker be successful in finding a host which permits mounting a filesystem, the consequences could be devastating; assign Attack lethality = 2.

8. Defensive Recommendation:

The potential target organization could protect itself by 1) using a firewall that blocks all services except only those needed for business or operational purposes, 2) checking systems to ensure they are not improperly configured (i.e. that only needed services are offered and configuration files are carefully set up), or 3) running vulnerability analysis tools to identify any errant services that are being unwittingly offered.

9. Multiple Choice Test Question:

What service is associated with port 635?

a. Mount
b. NFS
c. Linuxconf
d. Unknown

Answer: a

Detect 2

Server used for this query: [whois.arin.net]
St Lawrence College (NET-SLCSL)
2288 Parkdale Avenue Brockville Ontario, 5X3 CA
Netname: SLCSL
Netblock: 142.155.0.0 - 142.155.255.255

```
Sep 14 12:35:55 hostl proftpd[19393] hostl
  (142.155.41.179[142.155.41.179]): connected - local : a.b.c.159:21
Sep 14 12:35:55 hostl proftpd[19393] hostl
  (142.155.41.179[142.155.41.179]): connected - remote : 142.155.41.179:2398
Sep 14 12:35:55 hostl proftpd[19393] hostl
  (142.155.41.179[142.155.41.179]): received: USER anonymous
Sep 14 12:35:55 hostl proftpd[19393] hostl
  (142.155.41.179[142.155.41.179]): received: PASS (hidden)
Sep 14 12:35:55 hostl proftpd[19393] hostl
  (142.155.41.179[142.155.41.179]): ANON anonymous: Login successful.
Sep 14 12:35:56 hostl proftpd[19393] hostl
  (142.155.41.179[142.155.41.179]): received: REST 0
Sep 14 12:35:56 hostl proftpd[19393] hostl
  (142.155.41.179[142.155.41.179]): received: SYST
Sep 14 12:35:56 hostl proftpd[19393] hostl
  (142.155.41.179[142.155.41.179]): received: PWD
Sep 14 12:35:56 hostl proftpd[19393] hostl
  (142.155.41.179[142.155.41.179]): received: PASV
Sep 14 12:35:56 hostl proftpd[19393] hostl
  (142.155.41.179[142.155.41.179]): Entering Passive Mode
(a.b.c,159,160,159).
Sep 14 12:35:56 hostl proftpd[19393] hostl
  (142.155.41.179[142.155.41.179]): received: TYPE I
Sep 14 12:35:58 hostl proftpd[19393] hostl
  (142.155.41.179[142.155.41.179]): received: SIZE /
Sep 14 12:35:58 hostl proftpd[19393] hostl
  (142.155.41.179[142.155.41.179]): received: MDTM /
Sep 14 12:36:55 hostl proftpd[19404] hostl
  (142.155.41.179[142.155.41.179]): FTP session closed.
```

1. Source of Trace

http://www.sans.org/y2k/091500.htm

2. Detect was generated by:

It would appear that these are entries made by the ftp program into a system log. While I am not familiar with the format of the log, it is very basic.

The format separated by space delimiters is:

Timestamp (assume local time); local host name; ftp server name and connection ID; local host name; source IP; description

3. Probability the source address was spoofed:

There is no chance that the source IP is spoofed as FTP is an interactive protocol. If the address was spoofed, then the FTP session would not be established since the TCP 3-way handshake would not be completed (i.e. the 2nd part of the handshake would go back to an IP who never initiated the FTP session)

4. Description of Attack:

This packet trace shows an anonymous FTP connection. Without collaborating traces such as tcpdump, the actual reply codes by the FTP server are not known. Nonetheless, we can hypothesize about the intent of the potential hacker. The first thing noted are some unusual commands sent by the apparent hacker. REST 0 is an option in the FTP command set which, if implemented, is used to restart an interrupted transfer at a specific point, in this case, at the beginning. Could the potential hacker be trying to fish for a previous data exchange?
The potential hacker then follows with a SYST command to do some reconnaissance about the ftp server. Perhaps this could then lead him to identify a vulnerability and corresponding exploit. PWD checks the current working directory to perhaps get an understanding of where he is in the target's file system. PASV is again an optional command, which we see the server accepts, and is used to request from the server a new data port vice the default port 20. Note, here we see a likely typo in the trace submitted - it is likely that the submitter means that port 160 is the new data port (at the server), but the command port remains port 21. The potential hacker then changes the data type for subsequent transfers to "image" or binary. He then tries another optional command that is bizarre for binary transfer - he indicates the apparent logical byte size of the data type. Since "I" implies a raw binary data, this would appear unnecessary. The FTP RFC 959 indicates that the recipient (i.e. the server in this case) knows the end of file by the FTP client closing the connection. The presence of a "/" is unexplained, but perhaps the potential hacker is trying to crash the server with an unexpected character. The final exchange is not a valid FTP command, so it could be the name of the file to be transferred, however this should be preceded by a "STORE" or "PUT" command. Perhaps it is another attempt to see how stable the ftp server is. Perhaps, it is a word that is known to crash certain ftp implementations. In my view, this is the general intent of the potential hacker in this case.

5. Attack Mechanism:

The mechanism for this potential attack is FTP and is likely an attempt to "crash" the FTP server through unusually crafted commands. That the source IP belongs to a college is also a suggestion, albeit a stereotype, that this was a student who was playing.

6. Correlations:

This detect was attributed to Laurie@.edu

7. Evidence of Active Targeting:

There is no evidence from this single trace that indicates that this is active targeting.

8. Severity:

Typically, the following formula is used to calculate the severity of the attack. Points are assigned on a five point scale, 5 being the highest.

Severity = (Target criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures)

Target criticality - the target is being focused specifically and it is an ftp server; assign target criticality = 3.
Attack Lethality: There is nothing specifically lethal in this reconnaissance, however, it appears that the ftp server is being set up; assign Attack lethality = 2.
System Countermeasures - anonymous ftp is allowed so this is probably a public ftp server. Logging is enabled which is positive; assign system countermeasures = 3
Network countermeasures - it appears that this is a public ftp outside the firewall; assign network countermeasures=1

Severity = (3+2)-(3+1) = 1

9. Defensive Recommendation:

The potential target organization could protect itself by 1) using a firewall that blocks FTP, 2) installing tcpwrappers or configuring the firewall to allow FTP requests from specific addresses, or 3) considering carefully the need for the ftp service; and 4) ensuring, if needed, that default configurations are locked appropriately (i.e. anonymous or guest accounts are restricted).

10. Multiple Choice Test Question:

What are two types of FTP connections?

a. Active and Passive
b. Active and Restricted
c. Adaptive and Passive
d. Dynamic and Static

Answer: a

Detect 3

```
Server used for this query: [ whois.ripe.net ]
inetnum: 212.252.0.0 - 212.253.255.255
netname: TR-SUPERONLINE-980319
descr: Provider Local Registry
country: TR
```

```
Sep 2 21:38:13 212.253.18.156:3893 -> a.b.c.121:27374 SYN **S*****
Sep 2 21:38:13 212.253.18.156:3915 -> a.b.c.143:27374 SYN **S*****
Sep 2 21:38:17 212.253.18.156:3942 -> a.b.c.170:27374 SYN **S*****
Sep 2 21:38:15 212.253.18.156:3958 -> a.b.c.186:27374 SYN **S*****
Sep 2 21:38:15 212.253.18.156:3964 -> a.b.c.192:27374 SYN **S*****
Sep 2 21:38:15 212.253.18.156:3976 -> a.b.c.204:27374 SYN **S*****
Sep 2 21:38:15 212.253.18.156:3979 -> a.b.c.207:27374 SYN **S*****
Sep 2 21:38:41 212.253.18.156:4728 -> a.b.f.188:27374 SYN **S*****
Sep 2 21:38:41 212.253.18.156:4729 -> a.b.f.189:27374 SYN **S*****
=-=-=-=-=-=-=-=-=-=-=
```

1. Source of Trace

http://www.sans.org/y2k/090500-1200.htm

2. Detect was generated by:

Sep 2 21:38:13 212.253.18.156:3893 -> a.b.c.121:27374 SYN **S*****

This trace appears to come from Snort. The format separated by space delimiters is:

Timestamp (assume local time); source IP and port; destination IP and port; TCP Flags

3. Probability the source address was spoofed:

It is doubtful that the source address is spoofed in this attack. Most likely the potential hacker is searching for a response on the common port associated with the trojan SubSeven 2.1. While this is quite a noisy trojan scan and is automated with a script (more to follow below), given the time of the attack (local time: a weekend in the evening), it is doubtful that it would be noticed by a system administrator until later. While the address may not be spoofed, it is likely that the attacker is using a compromised account – otherwise, he would be at risk of being easily traced.

4. Description of Attack:

The potential attacker is looking for a host infected with the SubSeven 2.1 trojan. This trojan, similar to other “backdoor” programs, such as Netbus, Backorifice, etc., allows a remote user to take over the target machine. The hacker is then able to manipulate the hardware and software on the target as he sees fit. For example, he could install a keyboard monitor for password grabbing, install a Distributed Denial of Service tool for later activation, or simply wipe the poor target owner’s hard drive. Subseven 2.1 first made its appearance in January 2000, but there were previous variants and more recent variants also. As is often the case with malicious code, there are number of aliases for the trojan; for example, Subseven is also known as BackDoor-EP. Subseven runs as a client-server application. The attacker runs the client and the target host runs the server program.

A number of security sites provide extensive details of the characteristics of this particular trojan. One such site, a leading anti-virus vendor, contains information at the following link: http://vil.nai.com/vil/virusChar.asp?virus_k=10566.

Note that it states the trojan is installed to “listen” by default on port 27374, but this is configurable and hence not a sure bet. However, for potential hackers “fishing” for the existence of the trojan, installation on a port other than the default port makes detection that much harder. The trojan affects Intel-equipped hosts running Microsoft Windows 95/98 and is typically disguised as a jpg or bmp image, or bundled with another executable program. The server itself is about 336KB in size. The image file is often obtained by email as an attachment or via a web download, but could be installed on the target machine in any fashion. The link provides detailed information about the malicious code including filenames, installation directories, etc, that is useful in identifying and removing the trojan. Most, if not all, anti-virus software products, if current in their virus signature files, will automatically detect this trojan and offer the user options to delete or quarantine the program.

“When run, two files are installed into the WINDOWS folder of the user's hard disk. These two files are the main server exe files, normally called "MSREXE.EXE", and a loader program normally called "RUN.EXE", "WINDOS.EXE" or "MUEEXE.EXE". These filenames are only the default names and can be changed by the trojan's configuration program. The main server exe file is identified as "BackDoor-G2.svr" or "BackDoor-G2.svr.gen". The loader program is identified as "BackDoor-G2.ldr". Two other files are associated with this trojan, the configuration program and the client program used to communicate with the main server program. These are identified as BackDoor-G2.cfg and BackDoor-G2.cli respectively. These files do not hook the operating system and may be safely deleted if detected on the system.”

Additional information can be obtained from other links such as:

Symantec - http://www.symantec.com/avcenter/venc/data/sub.seven.20.html
ISS’s X-Force - http://xforce.iss.net/alerts/vol-4_num-1.php#subseven
CERT/CC - http://www.cert.org/present/cert-overview-trends/tsld179.htm
SANS - http://www.sans.org/y2k/subseven.htm
CVE - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660

Note that the general category of trojans is identified by the Common Vulnerabilities & Exposures (CVE) dictionary candidate number:

> CAN-1999-0660 (under review)
> This is a candidate for inclusion in CVE. It must be reviewed and accepted
> by the CVE Editorial Board before it can be added into CVE. Therefore,
> this candidate may be modified or even rejected in the future.
>
> NameCAN-1999-0660 (under review)
> DescriptionA hacker utility or Trojan Horse is installed on a
> system, e.g. NetBus, Back Orifice, Rootkit, etc.
> References
> PhaseProposed (19990804)
> VotesACCEPT(3) Wall, Northcutt, Hill

5. Attack Mechanism:

The mechanism for this potential attack is FTP and is likely an attempt to “crash” the FTP server through unusually crafted commands. That the source IP belongs to a college is also a suggestion, albeit a stereotype, that this was a student who was playing.

6. Correlations:

This detect was attributed to (Laurie Zirkle)

The following table, provided by a work colleague, Guy Bruneau, shows port scans on his home machine. I have included this as it serves to show the growth in popularity of the Subseven trojan. Statistics such as these show its relative popularity and may be used to help persuade key decision-makers to make operational decisions (e.g. fast-track a change to a firewall to block a particular port).

port_scan2000.htm

7. Evidence of Active Targeting:

There is no evidence from this single trace that indicates that this is active targeting. It would appear that the potential hacker is simply looking for infected hosts.

8. Severity:

Typically, the following formula is used to calculate the severity of the attack. Points are assigned on a five point scale, 5 being the highest.

Severity = (Target criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures)

Target criticality - there is nothing that seems to imply specific targeting or that the hosts are critical hosts; assign target criticality=1
Attack Lethality: While the consequences of finding a Subseven trojan are severe, there is nothing indicating its sucesss; assign Attack lethality = 2.
System Countermeasures - since it appears that the trojan scan was unsuccessful, we can conclude that the host is protected or is at least not infected. Since the trojan scan reached the host, I assume that the host countermeasures are reasonable in this case; assign system countermeasures = 3.
Network Countermeasures - since the probe for a well-known trojan got through to the host, I am suspect about the perimeter defence; assign network countermeasures=1

Severity = (1+2) - (3+1) = -1

9. Defensive Recommendation:

The potential target organization could protect itself by 1) using a firewall that blocks the default trojan port, 27374, 2) installing server-based (i.e. mail, web) and/or host-based anti-virus software that is kept current with the vendor-provided latest virus signature files and patches, 3) introducing an education-awareness program about the risks of downloading programs or “disguised” programs and the dangers of running them. The education program should also include who to contact for more information or to report an infection or suspicious behaviour, the method of contact to be used, etc.

The education program is vital as hackers can try all kinds of tricks to have trusting users install the trojan. Here is an excerpt from Symantec’s site that describes one such trick.

An email with an attachment called "server.exe" was spammed to Japanese computer users. The attachment claimed to be an anti-virus program for a virus called Pinkworm, but it was actually a trojan called SubSeven 2.0 Server. The email was sent from a Japanese Hotmail account claiming to be from Microsoft Japan Service. The email requests the recipient to run the attachment called "server.exe", which will protect the computer from the Pinkworm virus. Please note that there is no virus called “Pinkworm.”

(Reference http://www.symantec.com/avcenter/venc/data/sub.seven.20.html)

10. Multiple Choice Test Question:

What is the default protocol and port for the SubSeven 2.1 trojan?

a. UDP 27374
b. TCP 31337
c. TCP 27374
d. UDP or TCP 12345

Answer: c

Detect 4

(Klaus Steding-Jessen)

Name: PPa59-ResaleIndianapolis6-5R7158.saturn.bbn.com
Address: 4.54.61.248
BBN Planet (NET-SATNET)
150 Cambridge Park Dr.
Cambridge, MA 02138, US
Netname: SATNET
Netblock: 4.0.0.0 - 4.255.255.255
Maintainer: BBNP
Scan for ftp server machines:

```
Sep 15 01:49:45 hostname ipmon[xxxxx]: 01:49:45.938476 xxx0
  @0:36 b 4.54.61.248,1034 -> a.b.51.3,21 PR tcp len 20 48 -S IN
Sep 15 01:49:46 hostname ipmon[xxxxx]: 01:49:46.771460 xxx0
  @0:36 b 4.54.61.248,1034 -> a.b.51.3,21 PR tcp len 20 48 -S IN
Sep 15 01:49:47 hostname ipmon[xxxxx]: 01:49:47.572140 xxx0
  @0:36 b 4.54.61.248,1034 -> a.b.51.3,21 PR tcp len 20 48 -S IN
Sep 15 01:49:48 hostname ipmon[xxxxx]: 01:49:48.340882 xxx0
  @0:36 b 4.54.61.248,1034 -> a.b.51.3,21 PR tcp len 20 48 -S IN
Sep 15 01:49:57 hostname kernel: IP fw-in deny eth0 TCP
  4.54.61.248:1047 a.b.51.16:21 L=48 S=0x00 I=49408 F=0x0040 T=114
Sep 15 01:50:00 hostname kernel: IP fw-in deny eth0 TCP
  4.54.61.248:1047 a.b.51.16:21 L=48 S=0x00 I=55296 F=0x0040 T=114
Sep 15 01:50:06 hostname kernel: IP fw-in deny eth0 TCP
  4.54.61.248:1047 a.b.51.16:21 L=48 S=0x00 I=61696 F=0x0040 T=114
Sep 15 01:50:18 hostname kernel: IP fw-in deny eth0 TCP
  4.54.61.248:1047 a.b.51.16:21 L=48 S=0x00 I=10241 F=0x0040 T=114
```

Once an anonymous ftp server was found, the attacker tried to create some directories. Based on their names and elapsed time between each attempt, we can assume this is an automated tool:

```
Sep 15 01:49:51 hostname ftpd[xxxxx]: connection from
  PPPa59-ResaleIndianapolis6-5R7158.saturn.bbn.com (4.54.61.248)
Sep 15 01:49:52 hostname ftpd[xxxxx]: ANONYMOUS FTP LOGIN FROM
  PPPa59-ResaleIndianapolis6-5R7158.saturn.bbn.com, guest@here.com
Sep 15 01:49:53 hostname ftpd[xxxxx]: mkdir pub/. 54122314p
Sep 15 01:49:55 hostname ftpd[xxxxx]: mkdir . 56122314p
Sep 15 01:49:56 hostname ftpd[xxxxx]: mkdir bin/. 57122314p
```

After this, there was another connection. This time it does not seem to be automated:

```
Sep 15 01:50:51 hostname ftpd[yyyyy]: connection from
  PPPa59-ResaleIndianapolis6-5R7158.saturn.bbn.com (4.54.61.248)
Sep 15 01:50:52 hostname ftpd[yyyyy]: ANONYMOUS FTP LOGIN FROM
  PPPa59-ResaleIndianapolis6-5R7158.saturn.bbn.com, anonymous@on.the.net
Sep 15 01:51:27 hostname ftpd[yyyyy]: mkdir pub/FreeBSD/doc/teml
```

---

1. Source of Trace

http://www.sans.org/y2k/092000-1400.htm

2. Detect was generated by:

```
Sep 15 01:49:51 hostname ftpd[xxxxx]: connection from
  PPPa59-ResaleIndianapolis6-5R7158.saturn.bbn.com (4.54.61.248)
```

The traces provided come from "Unix" syslogs for several hosts. The various syslog entries are made by several daemons, including FTP, ipfwadm (i.e. a firewall), and perhaps TCP Wrappers. The format for each is similar - an explanation of the ftpd log entry should be representative of all entries.

Timestamp (assume local time); name of target host (sanitized here), the program responsible for the log entry with its process ID sanitized; log entry with source IP and resolved name.

3. Probability the source address was spoofed:

It is unlikely that this source address is spoofed; however, it is quite probable that the source host used for this activity has been compromised and is being used as a launch point. The potential hacker has timed his activity to the very early hours of a weekday and it is unlikely that it would be noticed immediately.

4. Description of Attack:

The potential attacker is looking for an ftp server and doing some network reconnaissance. In the first trace, the attacker is scanning two hosts looking for the ftp port, port 21. What is interesting is that, assuming that this is not an abbreviated trace, the reconnaissance is very sparse - perhaps it will be under the radar screen of a system administrator. The log entries are also interesting and both appear to be crafted packets. The entry by ipmon contains "PR" that implies the PUSH/RST TCP flags were set. This is not a normal flag setting. Similarly, for the entry by ipfwadm, the flag settings are 0x0040 or RST. This could be an attempt to solicit a response from either the host or a firewall/router in front. But in all likelihood, given the fixation with attempting to create a new directory on the ftp server, the attacker is simply searching for an ftp site for a potential warez site.

It is apparent initially that a tool is being used. One can note the rapid repetition of individual scans and the correlation between source port and destination IP (the source port increments by 13 and the target IP increments by the same number), and the likely crafted TCP flags within the packets.

Continuing with the trace, the apparent attacker tries an anonymous connection. This is probably against host a.b.51.3 given the timestamp. The tool then tries to create directories in the public, current, and binary directories. Note that he tries to hide the new directories from the normal "ls" (i.e. list) command by using a leading period in the attempted directory names. The directory names are almost identical. A minute later, the attacker is back to attempt to make another directory storing FreeBSD. I disagree with the author that this is unlikely an automated script because of the proximity in time. In my opinion, once the ftp server was found, the second and third connections were automatically scheduled. The third connection is independent of the success of the second - rather, it is simply another attempt to make a directory. If successful, it is my guess that the site could be advertised as a possible warez site.

5. Attack Mechanism:

The mechanism for this potential attack is FTP and is likely an attempt to identify FTP servers or perform additional network mapping to better understand the organization's topology. Note that the potential attacker selected only a few hosts to query and didn't just try and connect to an ftp server by initiating an ftp connection request. The potential hacker is probably trying to be stealthy - in terms of time of the scan, the number of hosts scanned, and the perhaps the rate of scanning.

6. Correlations:

This detect was attributed to (Klaus Steding-Jessen)

In addition the following correlation two days later was noted:
(Ref: http://www.sans.org/y2k/092200.htm)

(Stephen McNall)
Correlation with (Klaus Steding-Jessen)
Not an intrusion as such
... clip ...

```
Sep 15 01:50:51 hostname ftpd[yyyyy]: connection from
  PPPa59-ResaleIndianapolis6-5R7158.saturn.bbn.com (4.54.61.248)
Sep 15 01:50:52 hostname ftpd[yyyyy]: ANONYMOUS FTP LOGIN FROM
  PPPa59-ResaleIndianapolis6-5R7158.saturn.bbn.com, anonymous@on.the.net
Sep 15 01:51:27 hostname ftpd[yyyyy]: mkdir pub/FreeBSD/doc/teml
```

... clip ...
I received spam e-mail tonight
... clip ...
Received: from
PPPa53-ResaleKansasCity1-4R7102.saturn.bbn.com_[4.4.205.146]
[4.4.205.146] by shengna.com (SMTPD32-6.00) id A2826E0184;
Wed, 20 Sep 2000 15:08:50 +0800
Received:
from net.seven.it by
PPPa53-ResaleKansasCity1-4R7102.saturn.bbn.com with ESMTP;
Wed, 20 Sep 2000 02:03:17 -0500
... clip ...
at least a bit interesting it seems.

The saga continues. Looking into the source IP a little further, it appears by doing a reverse DNS along with a ping, that this IP is static (i.e. it is always 4.54.61.248). This is in contrast with PPPa53-ResaleKansasCity1-4R7102 which when pinged returns a different address in the assigned IP block (i.e. 4.4.205.152) than the one listed in the trace.

One can also note that by correlating timezones, both email messages went out between 02:08 and 02:03 GMT -5 respectively. My hunch here is that both of these mail servers were used as relays for spam. Mail.shengna.com is running a version of Linux (2.2.14-12.4RS on an i586) which could have a sendmail vulnerability. Mr. McNall does not elaborate about the spam mail, but my conclusion is that the owner of the static IP or user of this compromised host is probably a script kiddie-type hacker.

7. Evidence of Active Targeting:

There is no evidence from this single trace that indicates this is active targeting. It would appear that the potential hacker is looking simply for ftp hosts for potential warez sites.

8. Severity:

Typically, the following formula is used to calculate the severity of the attack. Points are assigned on a 5 point scale, 5 being the highest.

Severity = (Target criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures)

Target criticality - the target is being focused specifically and it is an ftp server; assign target criticality = 3.
Attack Lethality: This appears to be a very persistent and focused attack on ftp servers; assign Attack lethality = 3.
System Countermeasures - anonymous ftp is allowed, so this is probably a public ftp server. Logging is enabled which is positive; assign system countermeasures = 3
Network countermeasures - it appears that this is a public ftp outside the firewall; assign network countermeasures=1

Severity = (3+3)-(3+1) = 2

9. Defensive Recommendation:

In this case, the potential targets are both misconfigured ftp and mail servers. Defensive recommendations would include 1) carefully configuring the operating system and servers to avoid being used as mules or warez sites, 2) remaining cognizant of vulnerabilities discovered in the server applications, and 3) reviewing logs frequently.

10. Multiple Choice Test Question:

How many full-duplex TCP connections are involved in an ftp session?

a. 1
b. 2
c. 4
d. 1 with one additional connection for each file transfer

Answer: d

Assignment 2

The tool to be demonstrated, Haktek v1.1 was downloaded from http://www.hackernet.de/

Haktek is an older generation hacker tool kit written in the 1997 timeframe for the MS-Windows environment. Haktek provides the following functionality: IP scanning, port scanning, configurable ping, mail bombing, anti-mail bombing, and finger. It also provides the user the option of saving a session to disk. HakTek has a simple GUI interface and basic instructions on how to use the tool. It is not as stealthy, flashy or feature-rich as some more modern tools, but it is simple, relatively fast, and, always a bonus, works. It comes in a zip file of 225KB and the executable weighs in at about 650KB. The tool can be used for reconnaissance as well as destructive purposes.

The tool was tested on my "home lab" network comprised of three machines: "a" (Windows 95), "b" (Linux Mandrake 7.1) and "c" (Windows 98). The tool was run on "a" against "b" and the following session was saved to disk. Please note that the annotations are in blue, tool output is in black, and the tcpdump trace is in red.

Session begins …
==================================

/* target changed to the Linux box

Target changed to 10.10.10.2.

/* initial test of the ping feature to ensure it works; set ping count to 3, size to 1024.

Screen shot here

```
Pinging 10.10.10.2, with 1024 bytes of data. Count=3...
Reply from 10.10.10.2, round trip time = 4 ms
Reply from 10.10.10.2, round trip time = 3 ms
Reply from 10.10.10.2, round trip time = 3 ms

Average trip = 3 ms
```

```
02:49:39.538655 a > b: icmp: echo request
02:49:39.538773 b > a: icmp: echo reply
02:49:39.544247 a > b: icmp: echo request
02:49:39.544303 b > a: icmp: echo reply
02:49:39.549569 a > b: icmp: echo request
02:49:39.549625 b > a: icmp: echo reply
02:49:44.530662 arp who-has a tell b
02:49:44.531037 arp reply a is-at 0:60:97:14:bf:3b
```

/* okay, now let's go back and hit the target with larger echo requests. Note round trip time goes up.

```
Pinging 10.10.10.2, with 25000 bytes of data. Count=3...
Reply from 10.10.10.2, round trip time = 46 ms
Reply from 10.10.10.2, round trip time = 46 ms
Reply from 10.10.10.2, round trip time = 46 ms

Average trip = 46 ms
```

/* as all traces are identical, only a trace from one count is shown

```
02:49:46.097511 a > b: icmp: echo request (frag 19321:1480@0+)  /* first ping, packet fragmented due to size
02:49:46.098750 a > b: (frag 19321:1480@1480+)                  fragment ID - 19321;  packet data transmitted
```

```
02:49:46.099988 a > b: (frag 19321:1480@2960+)              in 1480 byte increments (ICMP header is
02:49:46.101232 a > b: (frag 19321:1480@4440+)              responsible for other 20 bytes) within Ethernet
02:49:46.102449 a > b: (frag 19321:1480@5920+)              frame /*
02:49:46.103675 a > b: (frag 19321:1480@7400+)
02:49:46.104911 a > b: (frag 19321:1480@8880+)
02:49:46.106144 a > b: (frag 19321:1480@10360+)
02:49:46.107373 a > b: (frag 19321:1480@11840+)
02:49:46.108598 a > b: (frag 19321:1480@13320+)
02:49:46.109830 a > b: (frag 19321:1480@14800+)
02:49:46.111080 a > b: (frag 19321:1480@16280+)
02:49:46.112296 a > b: (frag 19321:1480@17760+)
02:49:46.113522 a > b: (frag 19321:1480@19240+)
02:49:46.114752 a > b: (frag 19321:1480@20720+)
02:49:46.116000 a > b: (frag 19321:1480@22200+)
02:49:46.117040 a > b: (frag 19321:1328@23680)
02:49:46.117447 b > a: (frag 4754:1328@23680)                 /* b is now echoing back to a.  Note the
02:49:46.118048 b > a: (frag 4754:1480@22200+)                packets are sent back out-of-order; ICMP
02:49:46.119157 b > a: (frag 4754:1480@20720+)                doesn’t care
02:49:46.120391 b > a: (frag 4754:1480@19240+)
02:49:46.121645 b > a: (frag 4754:1480@17760+)
02:49:46.122853 b > a: (frag 4754:1480@16280+)
02:49:46.124083 b > a: (frag 4754:1480@14800+)
02:49:46.125314 b > a: (frag 4754:1480@13320+)
02:49:46.126545 b > a: (frag 4754:1480@11840+)
02:49:46.127777 b > a: (frag 4754:1480@10360+)
02:49:46.129007 b > a: (frag 4754:1480@8880+)
02:49:46.130238 b > a: (frag 4754:1480@7400+)
02:49:46.131468 b > a: (frag 4754:1480@5920+)
02:49:46.132699 b > a: (frag 4754:1480@4440+)
02:49:46.133931 b > a: (frag 4754:1480@2960+)
02:49:46.135161 b > a: (frag 4754:1480@1480+)
02:49:46.136392 b > a: icmp: echo reply (frag 4754:1480@0+)   /* b completes the first ping sequence
                                                              with an echo request
02:49:46.147376 a > b: icmp: echo request (frag 19577:1480@0+) /* second ping sequence starts …
02:49:46.148603 a > b: (frag 19577:1480@1480+)
02:49:46.149835 a > b: (frag 19577:1480@2960+)
02:49:46.151081 a > b: (frag 19577:1480@4440+)
02:49:46.152298 a > b: (frag 19577:1480@5920+)
```

/* Now, let’s scan the host for services on the well-known ports

```
Scanning host 10.10.10.2, ports 0 to 1024

Port 21 found. Desc='ftp'
Port 23 found. Desc='telnet'
Port 98 found.
Port 110 found. Desc='pop3'
Port 113 found. Desc='auth'
Port 515 found. Desc='printer'

Done!
```

/* note that the scan is very noisy. The scan is repeated four times for each unsuccessful port. The first time, the tool walks up each target host port. Source IP ports are incremented also. The subsequent three scans are repeated, but this time the target host port is randomly selected. The delta between scanned ports (e.g. tcpmux (port 1), to echo (port 7) is matched by the delta change in the source IP ports. The tool determines the existence of a service by trying to establish a 3-way handshake TCP connection request. Successful connections are torn down with the FIN command at the conclusion of scanning the entire desired target port range.

/* blank lines inserted for clarity; first scan through is sequential. For simplicity, this trace represents a scan for ports from 1 to 25.

```
02:50:41.366748 a.3507 > b.tcpmux: S 21168892:21168892(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x48]
02:50:41.366906 b.tcpmux > a.3507: R 0:0(0) ack 21168893 win 0 [tos 0x48]
```

/* note sequential port numbers; Service not available, so target sends back RST/ACK.

```
02:50:41.370057 a.3508 > b.2: S 21168895:21168895(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x49]
02:50:41.370109 b.2 > a.3508: R 0:0(0) ack 21168896 win 0 [tos 0x49]

02:50:41.373465 a.3509 > b.3: S 21168899:21168899(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x4a]
02:50:41.373535 b.3 > a.3509: R 0:0(0) ack 21168900 win 0 [tos 0x4a]
02:50:41.376778 a.3510 > b.4: S 21168902:21168902(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x4b]
02:50:41.376830 b.4 > a.3510: R 0:0(0) ack 21168903 win 0 [tos 0x4b]
02:50:41.380025 a.3511 > b.5: S 21168905:21168905(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x4c]
02:50:41.380093 b.5 > a.3511: R 0:0(0) ack 21168906 win 0 [tos 0x4c]
02:50:41.398072 a.3512 > b.6: S 21168923:21168923(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x4d]
02:50:41.398122 b.6 > a.3512: R 0:0(0) ack 21168924 win 0 [tos 0x4d]
02:50:41.401385 a.3513 > b.echo: S 21168926:21168926(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x4e]
02:50:41.401453 b.echo > a.3513: R 0:0(0) ack 21168927 win 0 [tos 0x4e]
02:50:41.404613 a.3514 > b.8: S 21168930:21168930(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x4f]
02:50:41.404663 b.8 > a.3514: R 0:0(0) ack 21168931 win 0 [tos 0x4f]
02:50:41.408038 a.3515 > b.discard: S 21168933:21168933(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x50]
02:50:41.408107 b.discard > a.3515: R 0:0(0) ack 21168934 win 0 [tos 0x50]
02:50:41.411292 a.3516 > b.10: S 21168936:21168936(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x51]
02:50:41.411342 b.10 > a.3516: R 0:0(0) ack 21168937 win 0 [tos 0x51]
02:50:41.414668 a.3517 > b.systat: S 21168940:21168940(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x94]
02:50:41.414736 b.systat > a.3517: R 0:0(0) ack 21168941 win 0 [tos 0x94]
02:50:41.418104 a.3518 > b.12: S 21168943:21168943(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0xd0]
02:50:41.418154 b.12 > a.3518: R 0:0(0) ack 21168944 win 0 [tos 0xd0]
02:50:41.421447 a.3519 > b.daytime: S 21168947:21168947(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0xa0]
02:50:41.421499 b.daytime > a.3519: R 0:0(0) ack 21168948 win 0 [tos 0xa0]
02:50:41.424800 a.3520 > b.14: S 21168950:21168950(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0xdc]
02:50:41.424870 b.14 > a.3520: R 0:0(0) ack 21168951 win 0 [tos 0xdc]
02:50:41.428339 a.3521 > b.netstat: S 21168953:21168953(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x1c]
02:50:41.428394 b.netstat > a.3521: R 0:0(0) ack 21168954 win 0 [tos 0x1c]
```

```
02:50:41.432099 a.3522 > b.16: S 21168957:21168957(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0xfc]
02:50:41.432153 b.16 > a.3522: R 0:0(0) ack 21168958 win 0 [tos 0xfc]
02:50:41.438251 a.3523 > b.qotd: S 21168963:21168963(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x38]
02:50:41.438327 b.qotd > a.3523: R 0:0(0) ack 21168964 win 0 [tos 0x38]
02:50:41.459239 a.3524 > b.msp: S 21168984:21168984(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x74]
02:50:41.459303 b.msp > a.3524: R 0:0(0) ack 21168985 win 0 [tos 0x74]
02:50:41.463822 a.3525 > b.chargen: S 21168989:21168989(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0xb0]
02:50:41.463876 b.chargen > a.3525: R 0:0(0) ack 21168990 win 0 [tos 0xb0]
02:50:41.468417 a.3526 > b.ftp-data: S 21168994:21168994(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x58]
02:50:41.468470 b.ftp-data > a.3526: R 0:0(0) ack 21168995 win 0 [tos 0x58]
```

/* ftp is available. Three-way handshake occurs

```
02:50:41.472835 a.3527 > b.ftp: S 21168998:21168998(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x94]
02:50:41.473160 b.ftp > a.3527: S 1778530533:1778530533(0) ack 21168999 win 32120 <mss
1460,sackOK,timestamp 2342745[|tcp]> (DF)
02:50:41.473627 a.3527 > b.ftp: . ack 1 win 8760 <nop,nop,timestamp 195442 2342745> (DF) [tos 0x94]

02:50:41.477069 a.3528 > b.ssh: S 21169002:21169002(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0xd0]
02:50:41.477143 b.ssh > a.3528: R 0:0(0) ack 21169003 win 0 [tos 0xd0]
```

/* telnet is available too

```
02:50:41.480421 a.3529 > b.telnet: S 21169006:21169006(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0xc]
02:50:41.480485 b.telnet > a.3529: S 1772783529:1772783529(0) ack 21169007 win 32120 <mss
1460,sackOK,timestamp 2342745[|tcp]> (DF)
02:50:41.480928 a.3529 > b.telnet: . ack 1 win 8760 <nop,nop,timestamp 195442 2342745> (DF) [tos 0xc]
02:50:41.484448 a.3530 > b.24: S 21169010:21169010(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x48]
02:50:41.484519 b.24 > a.3530: R 0:0(0) ack 21169011 win 0 [tos 0x48]
02:50:41.487945 a.3531 > b.smtp: S 21169013:21169013(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x84]
02:50:41.488030 b.smtp > a.3531: R 0:0(0) ack 21169014 win 0 [tos 0x84]
```

/* now tear down ftp and telnet connections from attackers side

```
02:50:41.488828 a.3527 > b.ftp: F 1:1(0) ack 1 win 8760 <nop,nop,timestamp 195442 2342745> (DF) [tos 0x94]
02:50:41.488905 b.ftp > a.3527: . ack 2 win 32120 <nop,nop,timestamp 2342746 195442> (DF)
02:50:41.490426 a.3529 > b.telnet: F 1:1(0) ack 1 win 8760 <nop,nop,timestamp 195442 2342745> (DF) [tos 0xc]
02:50:41.490480 b.telnet > a.3529: . ack 2 win 32120 <nop,nop,timestamp 2342746 195442> (DF)
```

/* now start 2nd of four scan attempts, this time in random order. Note source IP port delta changes the same amount as target host port deltas.

```
02:50:41.838445 a.3509 > b.3: S 21168899:21168899(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x4a]
02:50:41.838715 b.3 > a.3509: R 0:0(0) ack 1 win 0 [tos 0x4a]
```

/* delta between port 3 and systat (port 11) is eight. Hence source port increments by 8.

```
02:50:41.838523 a.3517 > b.systat: S 21168940:21168940(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x94]
02:50:41.838886 b.systat > a.3517: R 0:0(0) ack 1 win 0 [tos 0x94]

02:50:41.838605 a.3513 > b.echo: S 21168926:21168926(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x4e]
02:50:41.838947 b.echo > a.3513: R 0:0(0) ack 1 win 0 [tos 0x4e]

02:50:41.838690 a.3514 > b.8: S 21168930:21168930(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x4f]
02:50:41.839057 b.8 > a.3514: R 0:0(0) ack 1 win 0 [tos 0x4f]
```

/* scan continues …

/* Now, lets do some network mapping and see which hosts are alive. This reconnaissance could be followed by a port scan of "alive" hosts. Note the tool uses icmp echo requests to determine the existence of a host.

```
Scanning from 10.10.10..1 to 10.10.10.32...    /* my internal lab IP range

a='xxxx'           /* xxxx and yyy are netbios and host names
b = 'yyy'
c

/* 3 hosts found!

Done.
```

```
02:51:06.281745 a > b: icmp: echo request
02:51:06.281867 b > a: icmp: echo reply
02:51:06.281825 arp who-has c tell a
02:51:06.381013 arp reply c is-at xx
02:51:06.481745 a > c: icmp: echo request
02:51:06.581867 c > a: icmp: echo reply
```

/* let's try the finger service specifically. As is typical with this tool, everything port scan is repeated four times.

```
Establishing real-time userlist... (Only works if the sysadmin is a moron)
---[ Finger session ]--------------------------------------------------------------------

02:51:57.588205 a.3534 > b.finger: S 21245136:21245136(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x5]
02:51:57.588315 b.finger > a.3534: R 0:0(0) ack 21245137 win 0 [tos 0x5]
02:51:58.091498 a.3534 > b.finger: S 21245136:21245136(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x5]
02:51:58.091584 b.finger > a.3534: R 0:0(0) ack 1 win 0 [tos 0x5]
02:51:58.626899 a.3534 > b.finger: S 21245136:21245136(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x5]
02:51:58.626964 b.finger > a.3534: R 0:0(0) ack 1 win 0 [tos 0x5]
02:51:59.176021 a.3534 > b.finger: S 21245136:21245136(0) win 8192 <mss 1460,nop,wscale
0,nop,nop,timestamp[|tcp]> (DF) [tos 0x5]
02:51:59.176093 b.finger > a.3534: R 0:0(0) ack 1 win 0 [tos 0x5]

====================================
```

Session ends

In sum, this is a noisy tool that reflects its age. It is not terribly sophisticated, but works and could be used for destructive as well as reconnaissance purposes. Note that the mail bombing and anti-mail bombing functions were not tested.

Assignment 3

References:
1. Practical Assignment, Lenny Zeltser, 15 Aug 2000 (http://www.sans.org/y2k/practical/Lenny_Zeltser.htm) was referred to for correlation.

2. Access database of the SNORT logs. Prepared by Jamie French, DND CIRT.

XXX Inc. has a requirement to assess its security posture and has requested that YYY Inc. provide a bid to provide security services for your facility. To assist YYY Inc in this purpose, two intrusion detection devices (IDS) (i.e. Snort) were installed for a month within the MY.NET network. A fairly standard rulebase was used as a security policy on the devices. Assistance from your staff was solicited to help determine the "best" overall location to install the devices.

General Points

It is apparent from even a cursory review of one-month's data furnished by the IDS equipment that XXX Inc's networks are being probed, reconnoitered, and attacked on a daily basis. This should not be surprising and was expected. It is commonly appreciated that being connected to the Internet as part of doing business (e.g. with partners, clients, suppliers, etc.) brings with it benefits and risks. XXX is not unique in this regard. A careful inspection of a number of specific observations will help demonstrate some key points.

Specific Observations

During the month of activity, a total of over 80,000 entries were logged by the IDSs. We can categorize some of the activity as reconnaissance, likely or actual attacks, and other activity. We will look at some examples from each category.

Reconnaissance activities

SYN/FIN Scan - Source IP 202.0.178.98 was responsible for 20040 SYN/FIN alerts against a significant portion of the MY.NET's network. Source and destination ports of 53 (DNS) were used to help pass the traffic through any perimeter defenses, such as firewalls or routers. The SYN/FIN TCP flags were set in each packet. The purpose of this scan is to determine your internal network topology by identifying which hosts reply to this malformed TCP packet. The overwhelming majority of the scan occurred on 28 Jun between 6:51 and 7:01 in the morning. A few more scans were detected between 29 Jun and 3 Aug. These later scans were more directed, primarily at MY.NET.1.3, 1.4, 1.5, 130.94, and 60.14, which implies a much more focused scan, perhaps to verify results from an earlier reconnaissance activity.

NMAP TCP Ping Scan - NMAP is a mapping tool written by Fyodor (confirm) that can also be used for mapping. On 27 Jun, and again, on 30 Jun, 8 and 27 July and 5 Aug, during non-work hours, a very slow-rate stealthy scan was used. The source port was set to 80 (the location of the web service) and the destination port was set to 53 (DNS), again to help ensure successful passage into the MY.NET network. The source IPs of the scan were 209.218.228.201 and .46. The rate of scanning was quite low - about four per minute repeated twice. This scan was also targeted specifically at one host, MY.NET.1.8.

Sun RPC High Port Access - There were 2311 alerts specifying "Attempted Sun RPC high port access." Of these attempts, 2229 were made by 205.188.153.111 that targeted port 32771 (ruserd) on MY.NET.217.126 from port 4000 from 3 Aug at 11:48 repeatedly until 5 Aug at 23:52. This port is often targeted by exploiting RPC vulnerabilities. Referencing Mr. Zeltser's earlier work, it would appear unlikely that given the repeated nature and duration of the alerts, that an attack is being attempted here. Mr. Zeltser has determined that the popular messaging system ICQ (ref: icq.mirabilis.com) uses servers that listen on port 4000. It would appear then that within the organization ICQ is being used by at least one host. A review of the corporate security policies would be needed to evaluate if this is appropriate use or not. There are many exploits also which target ICQ, so its continued use should be considered carefully.

Ref: Sam Spade (samspade.org), we can verify that the IP 205.188.153.111 is in fact an ICQ server with AOL.

Address Digger Results
(Version 3.1beta)

Let's go!
Official name: fes-d015.icq.aol.com

Addresses: 205.188.153.111

Queso Fingerprint - There were 11 alarms attributed over the period to ten different source IPs directed at eight different internal addresses in an attempt to obtain information about the operating system or service running on the host's destination port.

Ref: http://gil.di.uminho.pt/ftp/contrib/queso-980922b-1.i386.html

Queso identifies operating systems from the TCP packet signature
rather than banners, daemon versions, etc.
The current config file lists over 80 OS's and versions.
It can detect Linux Kernel versions and TCP responses from devices
such as routers, terminal servers, printers, etc.

Wingate Port 8080 Proxy Server - In total, there were 4214 + alerts identified as "Wingate 1080 or 8080 Attempts." Of this number, 3652 were directed against MY.NET.253.105. TCP 8080 is the http-proxy port and it is probable that the diverse number of source IPs were connecting to internal hosts via the web proxy. All told, there were 562 alerts against other internal IPs, most frequently among them, MY.NET.20.10, 97.101, and 99.85. In an organization as large as XXX Inc, it is entirely possible that there might be several web proxy servers.

SNMP Requests - 1080 alarms were raised about SNMP or Simple Network Management Protocol requests. SNMP can be used to set or get information from a host's MIB or management information base, and is often used to determine the health/status of key network equipment. All SNMP traffic was internally generated from typically MY.NET.97.X addresses against MY.NET.101.192. If a hacker was to have installed a sniffer inside the network, knowledge of the SNMP interest in 101.192 might invite further reconnaissance of that host.

Scans from Watchlisted IP addresses - From 02:14 27 Jun until 5 Aug, there were 4795 alerts identified as being from source IPs on the Watchlist 222 NET-NCFC. All source IPs were in the 159.226.X.X range. The destination IPs varied, but MY.NET.253.41-43, 100.65, 100.230, and 6.7 were prominent.
The destination ports were most often 25 (smtp) and 80 (web), indicating they were looking for mail or web servers. Both of these types of services have large numbers of vulnerabilities associated with them (e.g. cgi-scripts, sendmail, etc.). MY.NET.6.7 was also targeted for port 23 (Telnet). NET-NCFC is of interest because they belong to the Computer Network Center Chinese Academy of Sciences, and China is known to have an active in the Information Operations area. It could be that these internal IPs are mail servers and it is functioning normally. That is, other source IPs are sending mail to these mail servers, but are not being alerted because they are not on the watchlist. There was also 13976 alerts identified from source IPs on the Watchlist 220 -ILISDNNET990517. Destination addresses of interest are MY.NET.181.88 and ports of interest include ports 21 and 20, associated with ftp, and port 1088 (an ephemeral port). The source port is often 6699, which is associated with Napster.

Other Activity

Large UDP Packets - 1170 alerts identified as "Large UDP packets" were generated from source 211.40.176.214 against MY.NET.98.179 from 5 Aug at 18:30 to 18:59. This behaviour is noteworthy, but is localized in time. It is also at this same time a large number of ICMP Destination Unreachable (12313 alerts - 18:30-18:42) and PING-

ICMP Time-Exceeded (6690 alerts - 18:30 - 19:15) events, which implies that they are all probably related to a temporary network failure/problem.

Napster - There are 505 alerts involving Napster. These involve internal IPs MY.NET.97.204,229,230, 98.130 and 162.200 among a few others. It would appear that Napster traffic is moving bi-directionally. Again, a closer scrutiny of the security policies would indicate if this is appropriate use of corporate assets. Napster is also in a legal battle right now, and it would be potentially embarrassing to be caught up in that legal squabble as a site that is hosting a Napster server.

Potential compromise/ Possible Attacks

Wu-ftpd exploits - On approximately 16:34 30 Jun, IP 151.164.223.206 attempted an FTP exploit against MY.NET.99.16 and 144.59. This exploit is referenced as GIAC000623. These exploits were targeted against wu-ftp server, the ftp server from Washington University. Immediately thereafter, the same IP tried the site-exec exploit against the same internal IPs and also 100.165 and 156.127. Typically, ftp exploits attempt buffer overflows permitting the hacker to gain privileged access on the host.

Telnet and FTP Attempted Logins - On 5 Aug between 18:37 and 19:03, there were a number of attempted Telnet and FTP logins. These were all listed as having failed. Internal IPs in question were relatively few including MY.NET.6.7, 60.8, 60.11, 145.18. Of note also, is 99.51 is identified solely as having a Telnet daemon running. Of interest also, are the attempts mentioned earlier by the Chinese to target port 23 on 6.7. In none of these cases, was a log entry made that indicated that a login failure had occurred. Clearly, however, a Telnet daemon is running on 6.7, so how is this to be explained. The attempts to target as caught by the Watchlist NET-NCFC occurred on 4 Aug at 21:26. Further investigation of the host is required before a definitive answer as to whether the host was compromised might be offered.

Recommendations

The installation of the two IDSs and the review of their log data was performed as a first step in determining the general state of security risk to XXX Inc. To more comprehensively and accurately assess the nature of the risk, the following recommendations are suggested:

1. Better align the security policy of the IDSs to reflect XXX Inc's corporate security policy (e.g. is Napster or ICQ a permitted activity)
2. Install additional IDSs in order to capture more detailed traffic on all subnets of XXX Inc's network architecture.
3. Evaluate more carefully XXX Inc's network topology so as to ensure the best placement for locating the IDS equipment to meet the requirements of a more comprehensive study
4. Evaluate and update, if required, your security policy, procedures, training, and education awareness program to reflect the specific security threats/risks.
5. Based on a more comprehensive and tuned study, make specific recommendations on "hardening" your public and internal systems and architecture by considering firewalls, router access-control lists, proxies, anti-virus software, vulnerability assessments, etc.
6. Develop an executive-level briefing to raise awareness to the effects of network security on XXX Inc.'s business operations.

XYZ Inc. appreciates the offer to submit this bid and looks forward to hearing from you so as to schedule an early start time to the work.

Assignment 4

To complete assignment 3 presents a common problem to many security practitioners, such as security officers, system administrators, and intrusion detection analysts, namely, how to make sense of large datasets. In other words, how to sift through the data and find the needle in the haystack. As analysts are bombarded with more and more data from more and more systems (i.e. firewalls, syslogs, anti-virus logs, IDSs, routers, etc.), correlating the information becomes a very complex problem.

In assignment 3, there were over 80,000 records in the alert log and XXX records in the scan log. Making contextual sense of the data definitely can cause a headache. To aid in the task, I applied a number of basic manual procedures aided by some indispensable automated querying and sorting tools. Fortunately, as reference in assignment 3, a work colleague had collected all of the Snort logs into two MS-Access databases. However before diving into the "haystack", I formulated a basic plan of how to proceed. I decided to look first at the Alert log, the first thing that I attempted to do was to create a mental map of XXX Inc's network (i.e. MY.NET.X.X). I did this by creating a network topology map manually, by sorting the data on destination IP and scanning through the data. Here, I focused on dominant destination IPs; that is, were there IPs that were mentioned in the logs a large number of times. For each different subnet (i.e MY.NET.X.0), I would draw a cloud and put in any key IPs within that subnet annotating briefly why it was significant (e.g. a server of some type based on the destination port. In a short time, I had a general feeling that we were dealing with a class B-like address space for MY.NET domain and a feeling for some of the potential key servers. After creating my "network diagram", I summarized all "interesting" internal IP addresses for servers (e.g DNS, mail, ftp, telnet, web, proxy, etc) as well as any other IPs demonstrating an interesting service (e.g. Netbios, Napster, internal IPs.). Finally, I also added any questions (e.g. what service is on port X?) as a written reminder as they came to me, and identified on separate sheets of paper, facts that I gleaned as well, hypotheses about certain IP addresses, and finally an IP watchlist.

Second, I did a similar process this time sorting on the source IP. Again, I wanted to create a general view of the IP addresses that tended to repeat, and to become more familiar with the descriptions given by Snort. This is where it was easy to pick up the offending SYN/FIN scan source IP. I was also looking here for the internal MY.NET hosts acting as the source IP. Was the host communicating outside the network? If so, for what purpose? Could it have been compromised, etc.? If communicating internally, could it have been compromised and searching for other targets?

Armed with a general top view of the internal network and a grasp on the general noise, I then went through the data looking for key ports and changes in signature. To do this, I sorted the data this time on destination port and Snort signatures. Here I was trying to pay close attention to the possibility of just a few Snort alerts which might have escaped my birds-eye view. Key words to pay attention to here were "Login-failure", "exploit", tool names such as "nmap."

Now that my comfort level was increasing, I started to factor in the timestamps and correlate alerts. One interesting point was seeing IP 205.188.3.205 conduct a stealthy SYN/FIN scan on the 29 June, 30 June, and 1 Aug, only to also do a very noisy RPC scan on 29 June. This starts to create a mental picture of one of the attackers.

The next step was to continue filtering and parsing the data by formalizing queries. Rather than jumping between filters, at this step, you could begin to parse the one big dataset into smaller database tables. This made identifying tallies a bit easier.

The power of the MS-Access database was critical to sorting the data in different views and allowing the quick response to questions that popped up. The time to import and parse the data into key headings (i.e. timestamp, signature, source IP, source host, destination IP, destination host, etc.) and then present it in an analyst-friendly manner was essential. If time had permitted, it would have been interesting to have written some additional tools to have automatically tallied up the number of times a particular IP shows up, a particular port, or to automatically run the dataset through a network generator which could show the n:m relationship between IPs. It could even generate a network map for you and based on the data provide guesses (with degree of confidence) as to the type of host it is, or the type of operating system it is (e.g. netbios -> windows; rpc -> Unix), etc.

As useful as tools are however, there is no substitute for experience and discussion with peers. The old adage, “two heads are better than one” could never be truer. Different perspectives, levels of knowledge and experience, and intuition allow for a better scoping of the analysis task.