Global Information Assurance Certification Paper

GAIC Level Two Practicum
Intrusion Detection In Depth

Version 2.9 (22 May 2001)

SANS 2001 Baltimore, Maryland
13-20 May 2001

Garrott W. Christoph, Ph.D.

Table of Contents

Introduction and Network Description 3

Assignment 1 – Five Network Detects

- Detect 1: SYN Scan for Windows Net-Bios 5
- Detect 2: SYN Scan for Misconfigured FTP Servers 11
- Detect 3: SYN Scan for FTP interspersed with Net-Bios 25
- Detect 4: DNS SYN Scan 32
- Detect 5: UDP scan from a Possibly Trojaned Computer 37
  (Because these are too similar, and, because, I need the practice.)
- Detect 6: Looking for Sub Seven 45
- Detect 7: Possible WinGate Activity 51
- Detect 8: Trolling for Trojans (and maybe ftp) 57
- Bibliography 65

Assignment 2 – Describe State of Intrusion Detection

- Flag Burning 66
- Introduction 66
- Scanning 68
- TCP Communications and TCP Flags 69
- TCP Scans and OS Finger Printing Techniques 73
- tft.c 77
- Conclusion 83
- Appendix: Source Code for tft.c 83
- Bibliography 90

Assignment 3 – "Analyze This" 92

- Introduction 92
- Executive Summary 92
- Statistics 93
- Detects by Priority with Descriptions 101
- "Top Talkers" Analysis and Link Maps 110
- Defensive recommendations 126
- Analysis Procedure 127
- Appendix 130
- Bibliography 132

Introduction and Network Description

The network, that is the subject of this practicum, is an academic scientific research organization. Its size is equivalent to that of a university science department. For purposes of discussion, this organization will be referred to as mynet.org; and the logs

presented are sanitized to suggest this. Mynet.org is one many research and administrative elements in a larger entity referred to as the "organization" and which has been sanitized to myorg.org.

There are three features of academic environments that have bearing on this practicum. First, they are rich in intellectual resources, but devoid of informational assets of any direct monetary value whatsoever. So, academic environments do not attract the kind of hostile computer activity that, for example, a bank or credit card database might. On the other hand, publicly prominent scientific institutions such as this are often targets of hostile computer activity because the compromise of such institutions can provide a certain cachet within intruder circles. Additionally, institutions such as ours are sometimes targeted by organizations with a political, and often a quite unfriendly agenda.

In addition, academic environments do not operate under institutionally imposed top down management structures. Rather, individual, and intellectually entrepreneurial scientists budget, operate, and supervise their own laboratories, including computational infrastructure. The quality of computer security policy, implementation, practice, diligence, and awareness among laboratories varies between extraordinarily high to appallingly low. Unfortunately, there is much too much of the later and not nearly enough of the former. The level of computer security in scientific labs, is further compromised by the predisposition of laboratories to obtain cutting edge, specialized, high performance computer equipment, the computer security implications of which are ill-understood, and the playful predisposition of scientists and students, perhaps with insufficient training, to get their hands on new equipment to see how it works in detail.

Finally, academic organizations highly value academic freedom, the free exchange if ideas and information, open networks, and open source software. They are inherently and deeply resistant to firewalls, imposed security, rules, passwd aging, configuration policy, etc, etc (It takes a root compromise, to raise a village.)

This network consists of 2 class C subnets with nearly 400 assigned IP addresses. The network employs one gateway and several 10/100 Mbit switching hubs. It is predominately a heterogeneous Unix shop (20 some flavors and versions). But has a significant Windows 9x/NT/2000, Macintosh, and printer presence as well. We have one gateway, but an occasional modem springs up unbeknownst to anyone. There is a porous organizational firewall and IDS. The organizational IRT (not me or my lab) monitors and blocks intrusive IP addresses, internal workstations undertaking hostile activity, and all traffic through a few ports (especially napster and rpc requests). It is a very loose firewall, whose policy is in keeping with the academic values of the organization.

Mynet.org does take proactive steps to increase the level of computer security within the lab. We try (~85% successfully) to maintain root only with the two serious system administrators in the laboratory. We run an extensive shell script tool on every Unix computer every night to gather maintenance, configuration information, system status, as well as security related information. We TCP wrap all computers with deny "ALL : ALL"

rules plus exceptions and run inetd.conf daemons with "additional logging" parameters. The central syslog server, thus monitors for "refused connect from" elements that are used in turn to update automatically the hosts.deny file on the one computer through which we funnel all ingress and egress connectivity to the laboratory. We have installed ssh, and encourage (but not yet require) its use. We run crack on all passwds on a 365/24/7 basis. In practice, we have suffered 1-3 root compromises per year since 1994.

I have done "Let's Pretend IDS" by reading syslogs/messages for several years. About one month ago, I installed a "tcpdump/snort IDS" on 6 internal sensors on mynet.org. This IDS, cannot gather all traffic to and from mynet.org, rather it samples traffic by gathering broadcast traffic, all traffic through a few stand alone workstations, and all traffic through some critical servers. This new IDS has already yielded a large trove of new detects. These detects constitute the data used in Assignment 1 for this practicum.

Tcpdump binary files are collected in 24 hour segments, one for each sensor. At midnight, they are transferred to a limited access IDS analysis server, where snort is run on the files. Although I have made several modifications to it, and I anticipate further pruning of the rule set as my understanding evolves, for now I basically use all the rule sets provided by the www.snort.org site. As part of my daily computer chores, I look at the results of the daily scripting tools and now the snort results each morning on arrival at work

In the notes, the commands "geektools", "arin", "ripe", and "apnic" are aliases to "whois -h whois.geektools.com" etc.

The correct answers to multiple choice questions are indicated in bold

Assignment 1 – Five Network Detects

Detect 1: SYN Scan for Windows Net-Bios

Summary:

From: myorg.88.29 from varying but repeated ports.
To: mynet.org and port 139 (netbios)
Notice: Fast SYN scan
Monotonically increasing source port numbers
Repeat of probe to same dst IP addresses,
with different initial src ports,
But, with the same interval between src port numbers.
Possible (likely?) that all dst ports probed,
but IDS sensors are within mynet.org not
in front of mynet.org. So, not all packets captured.

Not Shown: Identical traffic from other inside the organization sites
myorg.76.89 2 scans
myorg.100.14 3 scans
myorg.88.29 1 scan

This IP address derives from inside my organization, but outside of my subnet, department, and area or responsibility.

Observed Traffic:

#---- grep myorg.88.29 portscan.log ----------

```
Jun 28 08:35:38 myorg.88.29:1965 -> mynet.5.167:139 SYN ******S*
Jun 28 08:35:38 myorg.88.29:1966 -> mynet.5.178:139 SYN ******S*
Jun 28 08:35:38 myorg.88.29:1969 -> mynet.5.166:139 SYN ******S*
Jun 28 08:35:41 myorg.88.29:1985 -> mynet.5.91:139 SYN ******S*
Jun 28 08:35:41 myorg.88.29:1990 -> mynet.5.196:139 SYN ******S*
Jun 28 08:35:43 myorg.88.29:1993 -> mynet.4.96:139 SYN ******S*
Jun 28 08:36:30 myorg.88.29:1965 -> mynet.5.167:139 SYN ******S*
Jun 28 08:36:30 myorg.88.29:1966 -> mynet.5.178:139 SYN ******S*
Jun 28 08:36:30 myorg.88.29:1969 -> mynet.5.166:139 SYN ******S*
Jun 28 08:36:32 myorg.88.29:1985 -> mynet.5.91:139 SYN ******S*
Jun 28 08:36:33 myorg.88.29:1990 -> mynet.5.196:139 SYN ******S*
Jun 28 08:36:34 myorg.88.29:1993 -> mynet.4.96:139 SYN ******S*
Jun 28 08:38:35 myorg.88.29:2129 -> mynet.5.195:139 SYN ******S*
Jun 28 08:38:36 myorg.88.29:2136 -> mynet.5.156:139 SYN ******S*
Jun 28 08:38:39 myorg.88.29:2172 -> mynet.5.74:139 SYN ******S*
Jun 28 08:38:41 myorg.88.29:2175 -> mynet.5.92:139 SYN ******S*
Jun 28 08:38:41 myorg.88.29:2176 -> mynet.5.148:139 SYN ******S*
Jun 28 08:38:45 myorg.88.29:2195 -> mynet.5.93:139 SYN ******S*
Jun 28 08:38:45 myorg.88.29:2197 -> mynet.5.52:139 SYN ******S*
```

#---- tcpdump –r dump_28June | grep angela4.mynet.org --------
 # All workstation specific tcpdumps look like this.
 # It seems odd to me that there are no recorded
 # probes to Unix boxes.
 # This is a SYN then PSH-ACK, clearly we are missing parts.
 # I have no IDS sensors on PCs. Sensors see broadcasts only.
08:36:30.032278 < myorg.88.29.1965 > angela4.mynet.org.netbios-ssn: S 4468865:4468865(0) win 8192 <mss 1460> (DF) (ttl 125, id 56276)

08:46:32.887982 < myorg.88.29.1965 > angela4.mynet.org.netbios-ssn: P 4470454:4470493(39) ack 108354089 win 7869>>> NBT (DF) (ttl 125, id 61985)

#---- tcpdump –r dump_28June | grep ziffer3.mynet.org --------

08:38:45.494303 < myorg.88.29.2195 > ziffer3.mynet.org.netbios-ssn: S 4599335:4599335(0) win 8192 <mss 1460> (DF) (ttl 125, id 2540)

08:48:48.149916 < myorg.88.29.2195 > ziffer3.mynet.org.netbios-ssn: P 4600544:4600583(39) ack 853803629 win 8019>>> NBT (DF) (ttl 125, id 35122)

. . . . 194 qualitatively similar records . . . .

1) Source of Trace:
My Laboratory

2) Detect was Generated by:
Site maintains 6 IDS sensors running tcpdump 24/7.

```
% tcpdump –w dumpFile_24hour
% snort –r dumpFile_24hour -c /etc/snort/snort.conf
```

At midnight, on a designated analysis workstation, process data using several shell, sed, awk, and perl scripts.
Gather all outside IP addresses and suspicious inside IP address of interest.
Obtain tcpdump data in separate files, for each "IP address of interest".

```
% tcpdump –r dump_24hour -vv "dst $outsider or src $outsider" \
        > tcpdump_IPaddress
% tcpdump -r dumpFile_24hour –vv -x \
        "dst $outsider or src $outsider" > tcpdump_hex_IPaddress
```

Snort Rule that generated this detect:
preprocessor portscan:

3) Probability that Source Address was Spoofed:
Unlikely.
Reason: This was a scan looking for netbios session service (open shares). The source

computer must be able to accept the response.

4) Description of Attack:

From:	myorg.88.29 from varying but repeated ports.
To:	mynet.org and port 139 (netbios)
Average Interval:	9.8 sec
Protocol:	TCP
ID Numbers:	Vary normally
Seq. Numbers:	Vary normally
TTL:	125

This incident was a stimulus event.
This incident involved only "Normal TCP/IP" traffic.
This incident occurred at moderate speed (6.1 hits/minute).
This incident involved a modest volume of traffic. It encompassed only 2 subnets only. .
This incident was a reconnaissance.
The target OS for this incident was windows/9x/NT/2000.
The source OS for this incident appears to be another windows box, perhaps a windows based server.

5) Attack Mechanism:

It is possible, that this is benign TCP/IP traffic from one part of the organization to another. But, it is a SYN scan, and elements of this organization should not be doing this against other elements. I have chosen to use this detect, because I did not fully understand what was going on. Also because, this detect, has bearing on detect 3, and I will be referring to it at that time.

It is **not** the consequence of a user browsing the "Network Neighborhood" in domains, not their own. We permit, even encourage this activity in the organization. I was unable to reproduce this activity by doing a "Network Neighborhood" browse against my monitored network.

At very least this is a scripted nbstat –a mynet.org.255. While this is not strictly against the "rules", I view it as at least moderately unfriendly.

Typically windows netbios uses these ports:
Port 137 Netbios name service
Communication 137 <-> 137
Port 138 Netbios datagram service (UDP-like)
Communication Ephemeral <-> 138
Port 139 Netbios session service (TCP-like)
Communication Ephemeral <-> 139

All these communications are directed from the *.myorg.org to mynet.255:139 (session

service). The packets are looking for Windows services. This is most likely a "Share Enumeration" by myorg.88.29 against mynet.org.

Name of tool: Perhaps the Windows command: nbtstat –a mynet.org.255

This incident has as it purpose the enumeration of open shares on Windows 9x boxes. Northcutt et al. pp 156 (2001). If a Windows box permits shares of disk drives or other resources, then the hostile can read, perhaps write to, or other wise alter and control the contents on the disk. Passwds can be gathered for off site cracking, thus the future integrity of the workstation can be compromised even when the share is removed, and possibly sensitive information can be obtained from the computer. If write permissions are granted the user also opens the computer to use as a repository of illegal software or worse. Open shares are used by the virus community as a method of installing viruses directly onto the computer without going through the uncertain process of sending a *.vbs e-mail attachment which might not be opened, or might be blocked by the mail exchange's antivirus protection (http://www.cert.org/incident_notes/IN-2000-02.html, http://www.cert.org/vul_notes/VN-2000-03.html). By spoofing the IP address of the hostile source, this method can be used to DOS a Windows box, by allowing a remote attacker to change a file sharing service, making it return an unknown driver type. This will cause the target to crash (CVE-2000-1003, http://www.cert.org/vul_notes/VN-2000-03.html).

Having said all that there are 2 other possibilities:

1) This is some kind of normal inside the Organization traffic which I do not fully understand. (I don't think this is true). I called some of the users of these computers (they are all PCs). I got no useful information.
2) A real spoofing of myorg.88.29, by an outsider, or a previously trojaned myorg.88.29 is auto scanning. (Unlikely)

The target for this kind of probe is a Windows 9x/NT/2000 PC.

This is, at most, the reconnaissance phase of an attack, and so no damage was done. Rather, information was gathered for a later more directed assault on a specific Windows box.

6) Correlations:

There are several CVE's associated with netbio vulnerabilities (http://cve.mitre.org/cve/):

CVE-1999-0153
Windows 95/NT out of band (OOB) data denial of service through NETBIOS port, aka WinNuke.
CVE-1999-0288
Denial of servic
e in WINS with malformed data to port 137 (NETBIOS Name Service).

CVE-1999-0810
Denial of service in Samba NETBIOS name service daemon (nmbd).
CVE-2000-0347
Windows 95 and Windows 98 allow a remote attacker to cause a denial of service via a NetBIOS session request packet with a NULL source name.
CVE-2000-0673
The NetBIOS Name Server (NBNS) protocol does not perform authentication, which allows remote attackers to cause a denial of service by sending a spoofed Name Conflict or Name Release datagram, aka the "NetBIOS Name Server Protocol Spoofing" vulnerability.

CVE-2000-1003
NETBIOS client in Windows 95 and Windows 98 allows a remote attacker to cause a denial of service by changing a file sharing service to return an unknown driver type, which causes the client to crash.

Hacking Exposed, Scambray, McClure, and Kurtz, pages 46 and 59, Second Edition, Osborne/McGraw Hill (2001)

If the hostile was looking for Unix boxes with excessively shared exported file systems:
CAN-1999-0520: SMB shares with poor access control

Promiscuously shared Unix exports.
CAN-1999-0554: NFS exports to the world. (http://cve.mitre.org/cve/)

7) Evidence of Active Targeting:
This incident shows some of active targeting.
This incident scanned at least part of two entire subnets. It was aimed specifically at port 139 on Windows operating systems, but it relied on broadcast brute force to find them

8) Severity:
Severity = (Criticality + Lethality) – ($CounterMeasures_{System}$ + $CounterMeasures_{Network}$)
Severity = (1 + 3) - (3 +1)
Severity = 0

Comment: All Windows boxes in the Organization are scanned monthly for open shares. Users are "counseled", usually they respond appropriately. It must be said, however, that there are a minority of non-compliant users. So, I assign $CounterMeasures_{System} = 3$.

9) Defensive Recommendation:
Open shares and NFS world exported files systems are among the "Top-Ten Most Critical Security Vulnerabilities" http://www.sans.org/topten.htm.
Port 137, 138, and 139 should be blocked at the organization firewall.
However, unless this IP address was spoofed and spoofed from outside the organization, blocking at the firewall will not help in this case. (As the incident apparently derives from

inside to organization.)

10) Multiple Choice Test Question:

In the following detect, which of theses statements is true.

08:38:45.494303 < myorg.88.29.2195 > ziffer3.mynet.org.netbios-ssn: S 4599335:4599335(0) win 8192 <mss 1460> (DF) (ttl 125, id 2540)

a) This is an example of a mal-formed or "crafted packet".
b) The fact that the first two numbers in the "4599335:4599335(0)" element are identical renders this as an "Interesting packet" worthy of investigation.
c) The source port is 139 (netbios-ssn) and the target is reserved port 2195.
d) The Time To Live for this packet is anomalous.
e) The DF flag, indicates that there will be more fragments to follow.
f) Because the source and destination for windows netbios traffic is always the same, the fact that port 2195 is trying to communicate with port 139 (netbios-ssn) makes this packet anomalous.
g) None of the above.

Detect 2: SYN Scan for Misconfigured FTP Servers

I include this detect, because I see this activity almost every day. For several years, I have detected these incidents with a script that continuously grep-s the central syslog server, which then parses a sufficiently offending hostile to the /etc/hosts.deny file, within a few seconds of the first detect. My new tcpdump/snort IDS, affords the possibility of a more thorough study of this signature.

Summary:

From:	202.39.225.97 Increasing Ephemeral ports starting with 3623
To:	mynet.org and 21 (ftp)
Notice:	The targets are all ftp ports.

```
#----- geektools 202.39.225.97 -----------------
[whois.geektools.com]
Query:    202.39.225.97
Registry:  whois.apnic.net
Results:

inetnum:    202.39.128.0 - 202.39.255.255
netname:    HINET-TW
descr:      CHTD, Chunghwa Telecom Co.,Ltd.
descr:      Data-Bldg.6F, No.21, Sec.21, Hsin-Yi Rd.
descr:      Taipei Taiwan 100
country:    TW
```

Observed Traffic:

```
#---- grep 202.39.225.97 portscan.log --------------------------

Jul  4 08:21:34 202.39.225.97:3623 -> mynet.4.182:21 SYN ******S*
Jul  4 08:21:34 202.39.225.97:3625 -> mynet.4.184:21 SYN ******S*
Jul  4 08:21:34 202.39.225.97:3627 -> mynet.4.186:21 SYN ******S*
Jul  4 08:21:34 202.39.225.97:3640 -> mynet.4.199:21 SYN ******S*
Jul  4 08:21:34 202.39.225.97:3642 -> mynet.4.201:21 SYN ******S*
Jul  4 08:21:34 202.39.225.97:3643 -> mynet.4.202:21 SYN ******S*
Jul  4 08:21:34 202.39.225.97:3676 -> mynet.4.235:21 SYN ******S*
Jul  4 08:21:34 202.39.225.97:3683 -> mynet.4.242:21 SYN ******S*
Jul  4 08:21:35 202.39.225.97:3787 -> mynet.5.90:21 SYN ******S*
Jul  4 08:21:35 202.39.225.97:3796 -> mynet.5.99:21 SYN ******S*
Jul  4 08:21:35 202.39.225.97:3825 -> mynet.5.128:21 SYN ******S*
Jul  4 08:21:35 202.39.225.97:3837 -> mynet.5.140:21 SYN ******S*
Jul  4 08:21:35 202.39.225.97:3838 -> mynet.5.141:21 SYN ******S*
Jul  4 08:21:35 202.39.225.97:3858 -> mynet.5.161:21 SYN ******S*
Jul  4 08:21:35 202.39.225.97:3865 -> mynet.5.168:21 SYN ******S*
Jul  4 08:21:38 202.39.225.97:3884 -> mynet.5.187:21 SYN ******S*
Jul  4 08:21:38 202.39.225.97:3896 -> mynet.5.199:21 SYN ******S*
Jul  4 08:22:27 202.39.225.97:3623 -> mynet.4.182:21 SYN ******S*
Jul  4 08:22:27 202.39.225.97:3625 -> mynet.4.184:21 SYN ******S*
Jul  4 08:22:27 202.39.225.97:3627 -> mynet.4.186:21 SYN ******S*
```

Jul 4 08:22:27 202.39.225.97:3640 -> mynet.4.199:21 SYN ******S*
Jul 4 08:22:27 202.39.225.97:3642 -> mynet.4.201:21 SYN ******S*
Jul 4 08:22:27 202.39.225.97:3643 -> mynet.4.202:21 SYN ******S*
Jul 4 08:22:27 202.39.225.97:3683 -> mynet.4.242:21 SYN ******S*
Jul 4 08:22:27 202.39.225.97:3686 -> mynet.4.245:21 SYN ******S*
Jul 4 08:22:28 202.39.225.97:3787 -> mynet.5.90:21 SYN ******S*
Jul 4 08:22:28 202.39.225.97:3796 -> mynet.5.99:21 SYN ******S*
Jul 4 08:22:28 202.39.225.97:3825 -> mynet.5.128:21 SYN ******S*
Jul 4 08:22:28 202.39.225.97:3837 -> mynet.5.140:21 SYN ******S*
Jul 4 08:22:28 202.39.225.97:3838 -> mynet.5.141:21 SYN ******S*
Jul 4 08:22:28 202.39.225.97:3858 -> mynet.5.161:21 SYN ******S*
Jul 4 08:22:28 202.39.225.97:3865 -> mynet.5.168:21 SYN ******S*
Jul 4 08:22:31 202.39.225.97:3884 -> mynet.5.187:21 SYN ******S*
Jul 4 08:22:31 202.39.225.97:3896 -> mynet.5.199:21 SYN ******S*

#---- tcpdump –r dump_04July | grep 202.39.225.97 ----------
 # These are PCs with no IDS sensors. So no RST packet observed.

08:22:20.017523 < 202.39.225.97.3459 > maria.mynet.org.ftp: S 3023291116:3023291116(0) win 32120 <mss 1460,sackOK,timestamp 1396
64046[|tcp]> (DF) (ttl 38, id 19528)

08:22:20.020035 < 202.39.225.97.3464 > typhoon.mynet.org.ftp: S 3026893013:3026893013(0) win 32120 <mss 1460,sackOK,timestamp 13
9664046[|tcp]> (DF) (ttl 38, id 19533)

. . . . 95 qualitatively similar records . . . .

#---- This is traffic to one of the anonymous ftp servers ----------

 # First a 3-way TCP/IP connection handshake
08:22:23.318660 < 202.39.225.97.3564 > portal.mynet.org.ftp: S 3030255354:3030255354(0) win 32120 <mss 1460,sackOK,timestamp 139664377[|tcp]> (DF) (ttl 38, id 19917)
08:22:23.318885 < portal.mynet.org.ftp > 202.39.225.97.3564: S 4064808221:4064808221(0) ack 3030255355 win 10136 <nop,nop,timestamp 52272796 139664377,nop,[|tcp]> (DF) (ttl 255, id 12825)
08:22:23.563665 < 202.39.225.97.3564 > portal.mynet.org.ftp: . 1:1(0) ack 1 win 32120 <nop,nop,timestamp 139664402 52272796> (DF) (ttl 38, id 19937)

 # Then data are transferred
08:22:23.939080 < portal.mynet.org.ftp > 202.39.225.97.3564: P 1:34(33) ack 1 win 10136 <nop,nop,timestamp 52272858 139664402> (DF) (ttl 255, id 12826)
08:22:24.183619 < 202.39.225.97.3564 > portal.mynet.org.ftp: . 1:1(0) ack 34 win 32120 <nop,nop,timestamp 139664464 52272858> (DF) (ttl 38, id 19953)
08:22:24.183943 < 202.39.225.97.3564 > portal.mynet.org.ftp: F 1:1(0) ack 34 win 32120 <nop,nop,timestamp 139664464 52272858> (DF) (ttl 38, id 19954)
08:22:24.184106 < portal.mynet.org.ftp > 202.39.225.97.3564: . 34:34(0) ack 2 win 10136 <nop,nop,timestamp 52272883 139664464> (DF) (ttl 255, id 12827)
08:22:24.184953 < portal.mynet.org.ftp > 202.39.225.97.3564: P 34:71(37) ack 2 win 10136 <nop,nop,timestamp 52272883 139664464> (DF) (ttl 255, id 12828)
08:22:24.194301 < portal.mynet.org.ftp > 202.39.225.97.3564: F 71:71(0) ack 2 win 10136 <nop,nop,timestamp 52272884 139664464> (DF) (ttl 255, id 12829)
08:22:24.430440 < 202.39.225.97.3564 > portal.mynet.org.ftp: R 3030255356:3030255356(0)

win 0 (ttl 229, id 19957)
08:22:24.439102 < 202.39.225.97.3564 > portal.mynet.org.ftp: R 3030255356:3030255356(0) win 0 (ttl 229, id 19958)
08:22:23.318660 < 202.39.225.97.3564 > portal.mynet.org.ftp: S 3030255354:3030255354(0) win 32120 <mss 1460,sackOK,timestamp 139664377[|tcp]> (DF) (ttl 38, id 19917)
08:22:23.318885 < portal.mynet.org.ftp > 202.39.225.97.3564: S 4064808221:4064808221(0) ack 3030255355 win 10136 <nop,nop,timestamp 52272796 139664377,nop,[|tcp]> (DF) (ttl 255, id 12825)
08:22:23.563665 < 202.39.225.97.3564 > portal.mynet.org.ftp: . 1:1(0) ack 1 win 32120 <nop,nop,timestamp 139664402 52272796> (DF) (ttl 38, id 19937)
08:22:23.939080 < portal.mynet.org.ftp > 202.39.225.97.3564: P 1:34(33) ack 1 win 10136 <nop,nop,timestamp 52272858 139664402> (DF) (ttl 255, id 12826)
08:22:24.183619 < 202.39.225.97.3564 > portal.mynet.org.ftp: . 1:1(0) ack 34 win 32120 <nop,nop,timestamp 139664464 52272858> (DF) (ttl 38, id 19953)
08:22:24.183943 < 202.39.225.97.3564 > portal.mynet.org.ftp: F 1:1(0) ack 34 win 32120 <nop,nop,timestamp 139664464 52272858> (DF) (ttl 38, id 19954)
08:22:24.184106 < portal.mynet.org.ftp > 202.39.225.97.3564: . 34:34(0) ack 2 win 10136 <nop,nop,timestamp 52272883 139664464> (DF) (ttl 255, id 12827)
08:22:24.184953 < portal.mynet.org.ftp > 202.39.225.97.3564: P 34:71(37) ack 2 win 10136 <nop,nop,timestamp 52272883 139664464> (DF) (ttl 255, id 12828)

 # And the connection is closed
08:22:24.194301 < portal.mynet.org.ftp > 202.39.225.97.3564: F 71:71(0) ack 2 win 10136 <nop,nop,timestamp 52272884 139664464> (DF) (ttl 255, id 12829)
08:22:24.430440 < 202.39.225.97.3564 > portal.mynet.org.ftp: R 3030255356:3030255356(0) win 0 (ttl 229, id 19957)
08:22:24.439102 < 202.39.225.97.3564 > portal.mynet.org.ftp: R 3030255356:3030255356(0) win 0 (ttl 229, id 19958)

#---- Messages from the Central Syslog Server ----------

Jul 4 07:56:21 aloft.mynet.org ftpd[29085]: refused connect from 202.39.225.97
Jul 4 08:22:03 ochre.mynet.org ftpd[686371]: refused connect from 202.39.225.97
Jul 4 08:22:03 octal.mynet.org ftpd[267263]: refused connect from 202.39.225.97
Jul 4 08:22:03 ocelot.mynet.org ftpd[1078571]: refused connect from 202.39.225.97
Jul 4 08:06:56 allegro.mynet.org ftpd[28045]: refused connect from 202.39.225.97

. . . . 81 qualitatively similar records . . . .

#---- /etc/inetd.conf -> in.ftpd -dl -t 10 -> enables session logging
 # to /var/adm/messages

Jul 4 08:22:23 portal inetd[147]: ftp[17260] from 202.39.225.97 3564
Jul 4 08:22:23 portal ftpd[17260]: FTPD: connection from 202.39.225.97 at Wed Jul 4 08:22:23 2001
Jul 4 08:22:23 portal ftpd[17260]: <--- 220
Jul 4 08:22:23 portal ftpd[17260]: portal FTP server () ready.
Jul 4 08:22:24 portal ftpd[17260]: <--- 221
Jul 4 08:22:24 portal ftpd[17260]: You could at least say goodbye.

Summary (similar detect):

From: 61.76.221.185 Increasing ports starting with 4371
To: mynet.org and 21 (ftp)

Notice:

IP Address : 61.76.220.0-61.76.229.255
Network Name : KORNET-XDSL-PUSAN
Connect ISP Name : KORNET

Org Name : PUSAN NODE
State : PUSAN
Address : 75 4KA JUNGANGDONG JUNGKU

Observed Traffic:

#---- grep 61.76.221.185 portscan.log ----------

Jun 27 04:05:22 61.76.221.185:4371 -> mynet.4.122:21 SYN ******S*
Jun 27 04:05:22 61.76.221.185:4372 -> mynet.4.123:21 SYN ******S*
Jun 27 04:05:25 61.76.221.185:4448 -> mynet.4.199:21 SYN ******S*

. . . . 14 qualitatively similar records . . . .

#---- tcpdump –r dump_27June | grep 61.76.221.185 ----------

04:05:12.123492 < 61.76.221.185.4275 > cheers.mynet.org.ftp: S 1662583894:1662583894(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 101, id 5525)

04:05:12.136675 < 61.76.221.185.4280 > gmc1200.mynet.org.ftp: S 1662839276:1662839276(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 101, id 5530)

04:05:12.141617 < 61.76.221.185.4282 > lex136.mynet.org.ftp: S 1662920388:1662920388(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 101, id 5532)

. . . . 55 qualitatively similar records . . . .

#---- This is one of the anonymous ftp servers ----------

I am missing the initial 3-way TCP/IP connection handshake. This occurred at ~04:00 AM, which is the time of day in which we do the central tape backups. It is possible that the backup traffic caused some tcpdump packets to be dropped. In any event, this trace clearly shows TCP traffic passing between the 2 computers.

04:05:23.060848 < 61.76.221.185.4372 > portal.mynet.org.ftp: . 1:1(0) ack 1 win 16968 (DF) (ttl 101, id 6087)
04:05:23.514468 < portal.mynet.org.ftp > 61.76.221.185.4372: P 1:34(33) ack 1 win 9898 (DF) (ttl 255, id 42408)
04:05:24.329081 < 61.76.221.185.4372 > portal.mynet.org.ftp: P 1:17(16) ack 34 win 16935 (DF) (ttl 101, id 6138)
04:05:24.329210 < portal.mynet.org.ftp > 61.76.221.185.4372: . 34:34(0) ack 17 win 9898 (DF) (ttl 255, id 42409)
04:05:24.346162 < portal.mynet.org.ftp > 61.76.221.185.4372: P 34:79(45) ack 17 win 9898 (DF) (ttl 255, id 42410)
04:05:25.154387 < 61.76.221.185.4372 > portal.mynet.org.ftp: P 17:38(21) ack 79 win 16890 (DF) (ttl 101, id 6170)
04:05:25.174921 < portal.mynet.org.ftp > 61.76.221.185.4372: P 79:127(48) ack 38 win 9898 (DF) (ttl 255, id 42411)

04:05:26.012559 < 61.76.221.185.4372 > portal.mynet.org.ftp: P 38:49(11) ack 127 win 16842 (DF) (ttl 101, id 6253)
04:05:26.016248 < portal.mynet.org.ftp > 61.76.221.185.4372: P 127:156(29) ack 49 win 9898 (DF) (ttl 255, id 42412)
04:05:26.842861 < 61.76.221.185.4372 > portal.mynet.org.ftp: P 49:68(19) ack 156 win 16813 (DF) (ttl 101, id 6279)
04:05:26.848781 < portal.mynet.org.ftp > 61.76.221.185.4372: P 156:195(39) ack 68 win 9898 (DF) (ttl 255, id 42413)
04:05:27.694888 < 61.76.221.185.4372 > portal.mynet.org.ftp: P 68:82(14) ack 195 win 16774 (DF) (ttl 101, id 6290)
04:05:27.698638 < portal.mynet.org.ftp > 61.76.221.185.4372: P 195:237(42) ack 82 win 9898 (DF) (ttl 255, id 42414)
04:05:28.541673 < 61.76.221.185.4372 > portal.mynet.org.ftp: P 82:102(20) ack 237 win 16732 (DF) (ttl 101, id 6312)
04:05:28.546092 < portal.mynet.org.ftp > 61.76.221.185.4372: P 237:285(48) ack 102 win 9898 (DF) (ttl 255, id 42415)
04:05:29.389619 < 61.76.221.185.4372 > portal.mynet.org.ftp: P 102:118(16) ack 285 win 16684 (DF) (ttl 101, id 6338)
04:05:29.393205 < portal.mynet.org.ftp > 61.76.221.185.4372: P 285:329(44) ack 118 win 9898 (DF) (ttl 255, id 42416)
04:05:30.232338 < 61.76.221.185.4372 > portal.mynet.org.ftp: P 118:134(16) ack 329 win 16640 (DF) (ttl 101, id 6370)
04:05:30.235982 < portal.mynet.org.ftp > 61.76.221.185.4372: P 329:373(44) ack 134 win 9898 (DF) (ttl 255, id 42417)
04:05:32.977523 < portal.mynet.org.ftp > 61.76.221.185.4372: P 329:373(44) ack 134 win 9898 (DF) (ttl 255, id 42418)
tcpdump: pcap_loop: truncated dump file
04:05:33.048683 < 61.76.221.185.4372 > portal.mynet.org.ftp: P 118:134(16) ack 329 win 16640 (DF) (ttl 101, id 6532)
04:05:33.048829 < portal.mynet.org.ftp > 61.76.221.185.4372: . 373:373(0) ack 134 win 9898 (DF) (ttl 255, id 42419)
04:05:33.814336 < 61.76.221.185.4372 > portal.mynet.org.ftp: P 134:141(7) ack 373 win 16596 (DF) (ttl 101, id 6569)
04:05:33.818203 < portal.mynet.org.ftp > 61.76.221.185.4372: P 373:402(29) ack 141 win 9898 (DF) (ttl 255, id 42420)
04:05:34.636063 < 61.76.221.185.4372 > portal.mynet.org.ftp: P 141:160(19) ack 402 win 16567 (DF) (ttl 101, id 6593)
04:05:34.640317 < portal.mynet.org.ftp > 61.76.221.185.4372: P 402:441(39) ack 160 win 9898 (DF) (ttl 255, id 42421)
04:05:35.439448 < 61.76.221.185.4372 > portal.mynet.org.ftp: P 160:174(14) ack 441 win 16528 (DF) (ttl 101, id 6616)
04:05:35.443107 < portal.mynet.org.ftp > 61.76.221.185.4372: P 441:483(42) ack 174 win 9898 (DF) (ttl 255, id 42422)
04:05:36.241055 < 61.76.221.185.4372 > portal.mynet.org.ftp: F 174:174(0) ack 483 win 16486 (DF) (ttl 101, id 6635)
04:05:36.241181 < portal.mynet.org.ftp > 61.76.221.185.4372: . 483:483(0) ack 175 win 9898 (DF) (ttl 255, id 42423)
04:05:36.241845 < 61.76.221.185.4372 > portal.mynet.org.ftp: R 1669848882:1669848882(0) win 0 (DF) (ttl 101, id 6636)
04:05:36.242600 < portal.mynet.org.ftp > 61.76.221.185.4372: P 483:520(37) ack 175 win 9898 (DF) (ttl 255, id 42424)
04:05:37.027485 < 61.76.221.185.4372 > portal.mynet.org.ftp: R 1669848882:1669848882(0) win 0 (ttl 101, id 6646)
04:05:37.028110 < 61.76.221.185.4372 > portal.mynet.org.ftp: R 1669848882:1669848882(0)

win 0 (ttl 101, id 6647)

#---- Messages from the Central Syslog Server ----------

Jun 27 03:39:25 aloft.mynet.org ftpd[17536]: refused connect from 61.76.221.185

Jun 27 03:49:54 allegro.mynet.org ftpd[17670]: refused connect from 61.76.221.185

Jun 27 03:48:53 albedo.mynet.org ftpd[20968]: refused connect from 61.76.221.185

 # This one was not refused, it is an anonymous ftp server
Jun 27 04:05:23 portal.mynet.org in.ftpd[26837]: connect from 61.76.221.185

. . . . 73 additional “refused connect from” messages . . . .

#---- /etc/inetd.conf -> in.ftpd -dl -t 10 enables Logging -------
 # This is not nice!

Jun 27 04:05:23 portal inetd[143]: ftp[26837] from 61.76.221.185 4372
Jun 27 04:05:23 portal ftpd[26837]: FTPD: connection from 61.76.221.185 at Wed Jun 27 04:05:23 2001
Jun 27 04:05:23 portal ftpd[26837]: <--- 220
Jun 27 04:05:23 portal ftpd[26837]: portal FTP server () ready.
Jun 27 04:05:24 portal ftpd[26837]: FTPD: command: USER anonymous
Jun 27 04:05:24 portal ftpd[26837]: <--- 331
Jun 27 04:05:24 portal ftpd[26837]: Guest login ok, send ident as password.
Jun 27 04:05:25 portal ftpd[26837]: FTPD: command: PASS guest@here.com
Jun 27 04:05:25 portal ftpd[26837]: <--- 230
Jun 27 04:05:25 portal ftpd[26837]: Guest login ok, access restrictions apply.
Jun 27 04:05:26 portal ftpd[26837]: FTPD: command: CWD /pub/
Jun 27 04:05:26 portal ftpd[26837]: <--- 250
Jun 27 04:05:26 portal ftpd[26837]: CWD command successful.
Jun 27 04:05:26 portal ftpd[26837]: FTPD: command: MKD 010626170301p
Jun 27 04:05:26 portal ftpd[26837]: <--- 550
Jun 27 04:05:26 portal ftpd[26837]: 010626170301p: Permission denied.
Jun 27 04:05:27 portal ftpd[26837]: FTPD: command: CWD /public/
Jun 27 04:05:27 portal ftpd[26837]: <--- 550
Jun 27 04:05:27 portal ftpd[26837]: /public/: No such file or directory.
Jun 27 04:05:28 portal ftpd[26837]: FTPD: command: CWD /pub/incoming/
Jun 27 04:05:28 portal ftpd[26837]: <--- 550
Jun 27 04:05:28 portal ftpd[26837]: /pub/incoming/: No such file or directory.
Jun 27 04:05:29 portal ftpd[26837]: FTPD: command: CWD /incoming/
Jun 27 04:05:29 portal ftpd[26837]: <--- 550
Jun 27 04:05:29 portal ftpd[26837]: /incoming/: No such file or directory.
Jun 27 04:05:30 portal ftpd[26837]: FTPD: command: CWD /_vti_pvt/
Jun 27 04:05:30 portal ftpd[26837]: <--- 550
Jun 27 04:05:30 portal ftpd[26837]: /_vti_pvt/: No such file or directory.
Jun 27 04:05:33 portal ftpd[26837]: FTPD: command: CWD /
Jun 27 04:05:33 portal ftpd[26837]: <--- 250
Jun 27 04:05:33 portal ftpd[26837]: CWD command successful.
Jun 27 04:05:34 portal ftpd[26837]: FTPD: command: MKD 010626170309p
Jun 27 04:05:34 portal ftpd[26837]: <--- 550
Jun 27 04:05:34 portal ftpd[26837]: 010626170309p: Permission denied.

```
Jun 27 04:05:35 portal ftpd[26837]: FTPD: command: CWD /upload/
Jun 27 04:05:35 portal ftpd[26837]: <--- 550
Jun 27 04:05:35 portal ftpd[26837]: /upload/: No such file or directory.
Jun 27 04:05:36 portal ftpd[26837]: <--- 221
Jun 27 04:05:36 portal ftpd[26837]: You could at least say goodbye.
```

#---- Finally, snort alerts for an Anonymous FTP Connection -------

```
[**] INFO FTP anonymous FTP [**]
06/27-04:05:24.329081 61.76.221.185:4372 -> mynet.4.123:21
TCP TTL:101 TOS:0x0 ID:6138 IpLen:20 DgmLen:56 DF
***AP*** Seq: 0x6387DE84  Ack: 0x7198688F  Win: 0x4227  TcpLen: 20
```

1) Source of Trace:

My Laboratory

2) Detect was Generated by:

Site maintains 6 IDS sensors running tcpdump 24/7.

```
% tcpdump –w dumpFile_24hour
% snort –r dumpFile_24hour -c /etc/snort/snort.conf
```

At midnight, on a designated analysis workstation, process data using several shell, sed, awk, and perl scripts.
Gather all outside IP addresses and suspicious inside IP address of interest.
Obtain tcpdump data in separate files, for each "IP address of interest".

```
% tcpdump –r dump_24hour -vv "dst $outsider or src $outsider" \
        > tcpdump_IPaddress
% tcpdump -r dumpFile_24hour –vv -x \
        "dst $outsider or src $outsider" > tcpdump_hex_IPaddress
```

Snort Rule that generated this detect:
preprocessor portscan:

And

```
policy.rules:
        alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"INFO FTP anonymous FTP";
        content:"anonymous"; nocase; flags:A+;)
```

3) Probability that Source Address was Spoofed:

None
Reason: This was a scan of my entire network looking for anonymous ftp servers. He required feedback to his computer to know when one was found. He found an anonymous ftp server (actually he found 2, the second is not shown). He established a TCP connection. All this activity requires 2-way communication. This could not have been a spoofed address.

4) Description of Attack:

From:	202.39.225.97 Increasing Ephemeral ports starting with 3623
From:	61.76.221.185 Increasing ports starting with 4371
To:	mynet.org and 21 (ftp)
Average Interval:	1.7 seconds
Protocol:	TCP
ID Numbers:	Increment normally
Seq. Numbers:	Increment normally
Ack. Numbers:	Increment normally

These incidents were stimuli with normal TCP responses.
This incident involved no crafted packets rather it used "Normal TCP/IP" traffic.
This incident was moderately fast (> 35 hits/minute).
This incident involved a moderate volume of traffic. It encompassed two entire subnets, 4 buildings, and several other segments of the entire organization.
This incident was a reconnaissance and exploitation of vulnerability as discovered.
The target OS for this incident appears to be anonymous ftp servers, especially those with with opportunistic write permissions, or perhaps unpatched wu-ftp stires.
The source OS for this incident is likely a Unix box.

5) Attack Mechanism:

This was a SYN scan to port 21. Following discovery of ftp servers listening to port 21, ftp attempts were immediately made. Attempts were made to create directories, and deposit software. Had he been successful, he would have deposited possibly illegally obtained software, exotic and possibly illegal files (child pornography), but most likely he would have installed the tools that could be used to break into and compromise this or other computers.

6) Correlations:

A somewhat similar detect is described in Becky Bogle's "GAIC Certification Practical" from the 2001 New Orleans meeting. http://www.sans.org/giactc/Becky_Bogle_GCIA.doc
Her attack differs in that the initial scan was through SYN-FIN scans to port 21. As explained by Ms. Bogle, that probe was likely diagnostic of a scan that had as its purpose the gathering of FTP banner information, and subsequent attack of ftp servers with the raman worm. The purpose of this attack would have been root compromise through buffer overflow. The SYN scan on my network, was immediately followed by an attempt to deposit files to my ftp servers, following anonymous login by the well know user: guest@here.com. The strategy of the two detects, if not the goals, are qualitatively similar.

There are many CVEs associated with ftp exploits (http://cve.mitre.org/cve/):

CVE-1999-0017

FTP servers can allow an attacker to connect to arbitrary ports on machines other than the FTP client, aka FTP bounce.
CVE-1999-0035
Race condition in signal handling routine in ftpd, allowing read/write arbitrary files.
CVE-1999-0054
Sun's ftpd daemon can be subjected to a denial of service.
CVE-1999-0075
PASV core dump in wu-ftpd daemon when attacker uses a QUOTE PASV command after specifying a username and password.
CVE-1999-0079
Remote attackers can cause a denial of service in FTP by issuing multiple PASV commands, causing the server to run out of available ports.
CVE-1999-0080
wu-ftp FTP server allows root access via "site exec" command.
CVE-1999-0081
wu-ftp allows files to be overwritten via the rnfr command.
CVE-1999-0082
CWD ~root command in ftpd allows root access.
CVE-1999-0083
getcwd() file descriptor leak in FTP
CVE-1999-0097
The AIX FTP client can be forced to execute commands from a malicious server through shell metacharacters (e.g. a pipe character).
CVE-1999-0183
Linux implementations of TFTP would allow access to files outside the restricted directory.
CVE-1999-0185
In SunOS or Solaris, a remote user could connect from an FTP server's data port to an rlogin server on a host that trusts the FTP server, allowing remote command execution.
CVE-1999-0201
A quote cwd command on FTP servers can reveal the full path of the home directory of the "ftp" user.
CVE-1999-0202
The GNU tar command, when used in FTP sessions, may allow an attacker to execute arbitrary commands.
CVE-1999-0219
Buffer overflow in Serv-U FTP server when user performs a cwd to a directory with a long name.
CVE-1999-0256
Buffer overflow in War FTP allows remote execution of commands.
CVE-1999-0302
SunOS/Solaris FTP clients can be forced to execute arbitrary commands from a malicious FTP server.
CVE-1999-0349
A buffer overflow in the FTP list (ls) command in IIS allows remote attackers to conduct

a denial of service and, in some cases, execute arbitrary commands.
CVE-1999-0351
FTP PASV "Pizza Thief" denial of service and unauthorized data access. Attackers can steal data by connecting to a port that was intended for use by a client.
CVE-1999-0362
WS_FTP server remote denial of service through cwd command.
CVE-1999-0368
Buffer overflows in wuarchive ftpd (wu-ftpd) and ProFTPD lead to remote root access, a.k.a. palmetto.
CVE-1999-0432
ftp on HP-UX 11.00 allows local users to gain privileges.
CVE-1999-0457
Linux ftpwatch program allows local users to gain root privileges.
CVE-1999-0671
Buffer overflow in ToxSoft NextFTP client through CWD command.
CVE-1999-0707
The default FTP configuration in HP Visualize Conference allows conference users to send a file to other participants without authorization.
CVE-1999-0777
IIS FTP servers may allow a remote attacker to read or delete files on the server, even if they have "No Access" permissions.
CVE-1999-0789
Buffer overflow in AIX ftpd in the libc library.
CVE-1999-0838
Buffer overflow in Serv-U FTP 2.5 allows remote users to conduct a denial of service via the SITE command.
CVE-1999-0878
Buffer overflow in WU-FTPD and related FTP servers allows remote attackers to gain root privileges via MAPPING_CHDIR.
CVE-1999-0879
Buffer overflow in WU-FTPD and related FTP servers allows remote attackers to gain root privileges via macro variables in a message file.
CVE-1999-0880
Denial of service in WU-FTPD via the SITE NEWER command, which does not free memory properly.
CVE-1999-0914
Buffer overflow in the FTP client in the Debian GNU/Linux netstd package.
CVE-1999-0950
"Buffer overflow in WFTPD FTP server allows remote attackers to gain root access via a series of MKD and CWD commands that create nested directories."
CVE-1999-0955
Race condition in wu-ftpd and BSDI ftpd allows remote attackers gain root access via the SITE EXEC command.
CVE-1999-0997
wu-ftp with FTP conversion enabled allows an attacker to execute commands via a

malformed file name that is interpreted as an argument to the program that does the conversion, e.g. tar or uncompress.
CVE-2000-0015
CascadeView TFTP server allows local users to gain privileges via a symlink attack.
CVE-2000-0040
glFtpD allows local users to gain privileges via metacharacters in the SITE ZIPCHK command.
CVE-2000-0044
Macros in War FTP 1.70 and 1.67b2 allow local or remote attackers to read arbitrary files or execute commands.
CVE-2000-0131
Buffer overflow in War FTPd 1.6x allows users to cause a denial of service via long MKD and CWD commands.
CVE-2000-0150
Firewall-1 allows remote attackers to bypass port access restrictions on an FTP server by forcing it to send malicious packets which Firewall-1 misinterprets as a valid 227 response to a client's PASV attempt.
CVE-2000-0462
ftpd in NetBSD 1.4.2 does not properly parse entries in /etc/ftpchroot and does not chroot the specified users, which allows those users to access other files outside of their home directory.
CVE-2000-0514
GSSFTP FTP daemon in Kerberos 5 1.1.x does not properly restrict access to some FTP commands, which allows remote attackers to cause a denial of service, and local users to gain root privileges.
CVE-2000-0565
SmartFTP Daemon 0.2 allows a local user to access arbitrary files by uploading and specifying an alternate user configuration file via a .. (dot dot) attack.
CVE-2000-0573
The lreply function in wu-ftpd 2.6.0 and earlier does not properly cleanse an untrusted format string, which allows remote attackers to execute arbitrary commands via the SITE EXEC command.
CVE-2000-0577
Netscape Professional Services FTP Server 1.3.6 allows remote attackers to read arbitrary files via a .. (dot dot) attack.
CVE-2000-0587
The privpath directive in glftpd 1.18 allows remote attackers to bypass access restrictions for directories by using the file name completion capability.
CVE-2000-0636
HP JetDirect printers versions G.08.20 and H.08.20 and earlier allow remote attackers to cause a denial of service via a malformed FTP quote command.
CVE-2000-0640
Guild FTPd allows remote attackers to determine the existence of files outside the FTP root via a .. (dot dot) attack, which provides different error messages depending on whether the file exists or not.

CVE-2000-0644
WFTPD and WFTPD Pro 2.41 allows remote attackers to cause a denial of service by executing a STAT command while the LIST command is still executing.
CVE-2000-0674
ftp.pl CGI program for Virtual Visions FTP browser allows remote attackers to read directories outside of the document root via a .. (dot dot) attack.
CVE-2000-0676
Netscape Communicator and Navigator 4.04 through 4.74 allows remote attackers to read arbitrary files by using a Java applet to open a connection to a URL using the "file", "http", "https", and "ftp" protocols, as demonstrated by Brown Orifice.
CVE-2000-0717
GoodTech FTP server allows remote attackers to cause a denial of service via a large number of RNTO commands.
CVE-2000-0761
OS2/Warp 4.5 FTP server allows remote attackers to cause a denial of service via a long username.
CVE-2000-0813
Check Point VPN-1/FireWall-1 4.1 and earlier allows remote attackers to redirect FTP connections to other servers ("FTP Bounce") via invalid FTP commands that are processed improperly by FireWall-1, aka "FTP Connection Enforcement Bypass."
CVE-2000-0837
FTP Serv-U 2.5e allows remote attackers to cause a denial of service by sending a large number of null bytes.
CVE-2000-0856
Buffer overflow in SunFTP build 9(1) allows remote attackers to cause a denial of service or possibly execute arbitrary commands via a long GET request.
CVE-2000-0870
Buffer overflow in EFTP allows remote attackers to cause a denial of service via a long string.
CVE-2000-0871
Buffer overflow in EFTP allows remote attackers to cause a denial of service by sending a string that does not contain a newline, then disconnecting from the server.
CVE-2000-0875
WFTPD and WFTPD Pro 2.41 RC12 allows remote attackers to cause a denial of service by sending a long string of unprintable characters.
CVE-2000-0876
WFTPD and WFTPD Pro 2.41 RC12 allows remote attackers to obtain the full pathname of the server via a "%C" command, which generates an error message that includes the pathname.
CVE-2000-0943
Buffer overflow in bftp daemon (bftpd) 1.0.11 allows remote attackers to cause a denial of service and possibly execute arbitrary commands via a long USER command.
CVE-2000-1027
Cisco Secure PIX Firewall 5.2(2) allows remote attackers to determine the real IP address of a target FTP server by flooding the server with PASV requests, which includes the real

IP address in the response when passive mode is established.
CVE-2000-1182
WatchGuard Firebox II allows remote attackers to cause a denial of service by flooding the Firebox with a large number of FTP or SMTP requests, which disables proxy handling.
CVE-2001-0053
One-byte buffer overflow in replydirname function in BSD-based ftpd allows remote attackers to gain root privileges.
CVE-2001-0054
Directory traversal vulnerability in FTP Serv-U before 2.5i allows remote attackers to escape the FTP root and read arbitrary files by appending a string such as "/..%20." to a CD command, a variant of a .. (dot dot) attack.
CVE-2001-0138
privatepw program in wu-ftpd before 2.6.1-6 allows local users to overwrite arbitrary files via a symlink attack.
CVE-2001-0187
Format string vulnerability in wu-ftp 2.6.1 and earlier, when running with debug mode enabled, allows remote attackers to execute arbitrary commands via a malformed argument that is recorded in a PASV port assignment.
CVE-2001-0295
Directory traversal vulnerability in War FTP 1.67.04 allows remote attackers to list directory contents and possibly read files via a "dir *./../.." command.
CVE-2001-0318
Format string vulnerability in ProFTPD 1.2.0rc2 may allow attackers to execute arbitrary commands by shutting down the FTP server while using a malformed working directory (cwd).

7) Evidence of Active Targeting:
This incident evidenced active targeting.
The attack was not "aimed" at specific workstation.
It was aimed at a specific series of networks.
And it was aimed at a particular service or port (21, ftp).

8) Severity:
Severity = (Criticality + Lethality) – ($CounterMeasures_{System}$ + $CounterMeasures_{Network}$)
Severity = (4 + 4) - (5 + 1)
Severity = 2

Comment: The anonymous ftp servers in my laboratory are chrooted, TCP-wrapped, fully patched, run on modern operating systems, and are closely monitored, so I assign $CounterMeasures_{System} = 5$. I have had compromises in the distant past on ftp servers in my lab, but none since I went on a campaign to properly configure them all.

9) Defensive Recommendation:
Damages: No damage done.

This incident did penetrate the organizational firewall. On the other hand we maintain anonymous ftp sites for appropriate and necessary scientific communication between our scientists and the greater scientific community. This incident was automatically locked out of the ftp server within a few seconds of his first ftp attempt. Unfortunately, this was not before he attempted ftp access to the ftp server. (This happens about half of the time in these kinds of probes.)

It is not reasonable to block port 21 at the firewall. (It is not remotely within the realm of politically feasibility.) This service needs to be placed in a DMZ outside of a laboratory firewall.

10) Multiple Choice Test Question:

The syslogs/messages files can be used to correlate information about intrusive ftp activity. Consider the following trace from /var/adm/messages file as taken from this detect.

```
Jul  4 08:22:23 portal inetd[147]: ftp[17260] from 202.39.225.97 3564
Jul  4 08:22:23 portal ftpd[17260]: FTPD: connection from 202.39.225.97 at Wed Jul  4
08:22:23 2001
Jul  4 08:22:23 portal ftpd[17260]: <--- 220
Jul  4 08:22:23 portal ftpd[17260]: portal FTP server () ready.
Jul  4 08:22:24 portal ftpd[17260]: <--- 221
Jul  4 08:22:24 portal ftpd[17260]: You could at least say goodbye.
```

Which of the following statements is true.

a) On the workstation portal, the deamon ftpd runs continuously.
b) This trace indicates that the hostile successfully logged in as the user anonymous.
c) It is likely, that the hostile in this episode acquired the ftp banner on portal.
d) Within the element "ftpd[17260]", the fact the number 17260 remains the same in throughout this detect, is indicative of packet crafting.
e) If tcpdump traces were available for this detect, it would not show a completed 3 way TCP handshake connection, because no login occurred.

Detect 3: SYN Scan for FTP interspersed with Net-Bios

Summary:

From:	128.100.70.4	Ports 137 and Ephemeral ports
To:	mynet.org	Ports 137 (Netbios name service) and 21 (ftp)
Notice:	This is a simultaneous and interspersed scan for both Windows name servers and vulnerable ftp sites.	
	Broadcast queries to port 137.	
	UDP port 137 scan through entire network	
	FTP prot 21 scan through entire network	

```
[root@liar Process]# geektools 128.100.70.4
[whois.geektools.com]
Query:    128.100.70.4
Registry:  whois.arin.net
Results:
University of Toronto Computing and Communications (NET-TORONTO)
   255 Huron Street Room 350
   Toronto ON, ON M5S1C1
   CA

[root@liar Process]# nslookup 128.100.70.4

Non-authoritative answer:
4.70.100.128.in-addr.arpa       name = lphm.phm.utoronto.ca.
```

Observed Traffic:

#---- grep 128.100.70.4 portscan.log ----------

```
Jun 27 09:44:11 128.100.70.4:137 -> mynet.4.33:137 UDP
Jun 27 09:44:12 128.100.70.4:137 -> mynet.4.122:137 UDP
Jun 27 09:44:12 128.100.70.4:1816 -> mynet.4.184:21 SYN ******S*
Jun 27 09:44:13 128.100.70.4:137 -> mynet.5.52:137 UDP
Jun 27 09:44:13 128.100.70.4:1954 -> mynet.4.33:21 SYN ******S*
Jun 27 09:44:13 128.100.70.4:1968 -> mynet.4.235:21 SYN ******S*
Jun 27 09:44:14 128.100.70.4:137 -> mynet.5.127:137 UDP
Jun 27 09:44:14 128.100.70.4:137 -> mynet.5.140:137 UDP
Jun 27 09:44:14 128.100.70.4:137 -> mynet.5.144:137 UDP
Jun 27 09:44:14 128.100.70.4:137 -> mynet.5.148:137 UDP
Jun 27 09:44:14 128.100.70.4:137 -> mynet.5.69:137 UDP
Jun 27 09:44:14 128.100.70.4:137 -> mynet.5.93:137 UDP
Jun 27 09:44:14 128.100.70.4:137 -> mynet.5.99:137 UDP
Jun 27 09:44:14 128.100.70.4:2145 -> mynet.5.80:21 SYN ******S*
Jun 27 09:44:14 128.100.70.4:2172 -> mynet.4.122:21 SYN ******S*
Jun 27 09:44:14 128.100.70.4:2177 -> mynet.5.82:21 SYN ******S*
Jun 27 09:44:15 128.100.70.4:137 -> mynet.5.161:137 UDP
Jun 27 09:44:15 128.100.70.4:137 -> mynet.5.191:137 UDP
Jun 27 09:44:16 128.100.70.4:137 -> mynet.4.246:137 UDP
Jun 27 09:44:16 128.100.70.4:137 -> mynet.5.222:137 UDP
```

Jun 27 09:44:17 128.100.70.4:2402 -> mynet.4.246:21 SYN ******S*
Jun 27 09:45:02 128.100.70.4:137 -> mynet.4.33:137 UDP
Jun 27 09:45:03 128.100.70.4:1816 -> mynet.4.184:21 SYN ******S*
Jun 27 09:45:04 128.100.70.4:137 -> mynet.5.52:137 UDP
Jun 27 09:45:04 128.100.70.4:1954 -> mynet.4.33:21 SYN ******S*
Jun 27 09:45:04 128.100.70.4:1994 -> mynet.4.245:21 SYN ******S*
Jun 27 09:45:05 128.100.70.4:137 -> mynet.5.127:137 UDP
Jun 27 09:45:05 128.100.70.4:137 -> mynet.5.140:137 UDP
Jun 27 09:45:05 128.100.70.4:137 -> mynet.5.144:137 UDP
Jun 27 09:45:05 128.100.70.4:137 -> mynet.5.148:137 UDP
Jun 27 09:45:05 128.100.70.4:137 -> mynet.5.69:137 UDP
Jun 27 09:45:05 128.100.70.4:137 -> mynet.5.93:137 UDP
Jun 27 09:45:05 128.100.70.4:137 -> mynet.5.99:137 UDP
Jun 27 09:45:05 128.100.70.4:2145 -> mynet.5.80:21 SYN ******S*
Jun 27 09:45:05 128.100.70.4:2177 -> mynet.5.82:21 SYN ******S*
Jun 27 09:45:06 128.100.70.4:137 -> mynet.5.161:137 UDP
Jun 27 09:45:06 128.100.70.4:137 -> mynet.5.191:137 UDP
Jun 27 09:45:07 128.100.70.4:137 -> mynet.4.246:137 UDP
Jun 27 09:45:07 128.100.70.4:137 -> mynet.5.222:137 UDP
Jun 27 09:45:08 128.100.70.4:2402 -> mynet.4.246:21 SYN ******S*

#---- tcpdump from src 128.100.70.4 ----------

 # netbios name server requests
09:44:08.490204 < lphm.phm.utoronto.ca.netbios-ns > lex136.mynet.org.netbios-ns:NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST

09:44:09.853253 < lphm.phm.utoronto.ca.netbios-ns > susan.mynet.org.netbios-ns:NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST

09:44:09.976086 < lphm.phm.utoronto.ca.netbios-ns > lex136.mynet.org.netbios-ns:NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST

 # SYN and RST to Possible Unix ftp site (tcp-wrapped)
09:44:13.062039 < lphm.phm.utoronto.ca.1968 > sunder.mynet.org.ftp: S 29189079:29189079(0) win 8192 <mss 1460> (DF) [tos 0x10](ttl 112, id 24165)

09:44:13.062080 < sunder.mynet.org.ftp > lphm.phm.utoronto.ca.1968: R 0:0(0) ack 29189080 win 0 (DF) [tos 0x10] (ttl 112, id 50461)

. . . . 154 qualitatively similar records . . . .

#---- grep lphm.phm.utoronto.ca /etc/syslog ----------

 # All these Unix boxes are tcp-wrapped
Jun 27 09:44:48 ocarina.mynet.org ftpd[101486]: refused connect from lphm.phm.utoronto.ca
Jun 27 09:28:40 albedo.mynet.org ftpd[21291]: refused connect from lphm.phm.utoronto.ca
Jun 27 09:29:41 allegro.mynet.org ftpd[17836]: refused connect from lphm.phm.utoronto.ca

 # portal is the anonymous ftp server, (thus allows off site ftp)
Jun 27 09:45:01 portal.mynet.org in.ftpd[432]: connect from lphm.phm.utoronto.ca

. . . . 96 additional “refused connect from” messages . . . .

#----- egrep ’128.100.70.4|432’ /var/adm/messages ----------

```
Jun 27 09:45:00 portal inetd[141]: ftp[432] from 128.100.70.4 1651
Jun 27 09:45:02 portal ftpd[432]: FTPD: connection from lphm.phm.utoronto.ca at Wed Jun 27
09:45:02 2001
Jun 27 09:45:02 portal ftpd[432]: <--- 220
Jun 27 09:45:02 portal ftpd[432]: portal FTP server () ready.
Jun 27 09:45:02 portal ftpd[432]: FTPD: command:
Jun 27 09:45:02 portal ftpd[432]: <--- 500
Jun 27 09:45:02 portal ftpd[432]: '': command not understood.
Jun 27 09:45:02 portal ftpd[432]: FTPD: command:
Jun 27 09:45:02 portal ftpd[432]: <--- 500
Jun 27 09:45:02 portal ftpd[432]: '': command not understood.
Jun 27 09:45:02 portal ftpd[432]: FTPD: command:
Jun 27 09:45:02 portal ftpd[432]: <--- 500
Jun 27 09:45:02 portal ftpd[432]: '': command not understood.
Jun 27 09:45:02 portal ftpd[432]: <--- 221
Jun 27 09:45:02 portal ftpd[432]: You could at least say goodbye.
```

Summary (Second Similar Detect)

From:	128.104.35.50	Ports 137 and Ephemeral ports
To:	mynet.org	Ports 137 (Netbios name service) and 21 (ftp)
Notice:	This is a simultaneous and interspersed scan for both Windows name servers and vulnerable ftp sites.	
	Unicast queries to port 137.	
	UDP port 137 scan through entire network	
	FTP prot 21 scan through entire network	

```
[root@liar Process]# geektools 128.104.35.50
[whois.geektools.com]
Query:    128.104.35.50
Registry:  whois.arin.net
Results:
University of Wisconsin-Madison (NET-WISC-HERD)
   1210 West Dayton Street
   Madison, WI 53706
   US
[root@liar Process]# nslookup 128.104.35.50

50.35.104.128.in-addr.arpa      name = soybean.agronomy.wisc.edu.
```

Observed Traffic:

#---- grep 128.104.35.50 portscan.log ----------

```
Jun 27 18:12:00 128.104.35.50:137 -> mynet.4.96:137 UDP
Jun 27 18:12:02 128.104.35.50:137 -> mynet.4.23:137 UDP
```

Jun 27 18:12:02 128.104.35.50:3321 -> mynet.4.184:21 SYN ******S*
Jun 27 18:12:02 128.104.35.50:3322 -> mynet.4.182:21 SYN ******S*
Jun 27 18:12:03 128.104.35.50:137 -> mynet.4.122:137 UDP
Jun 27 18:12:03 128.104.35.50:3487 -> mynet.4.23:21 SYN ******S*

. . . . 45 qualitatively similar records . . . .

#---- tcpdump from src 128.104.35.50 ----------

Windows netbios name server requests (note Unicasts)
18:11:59.267983 < soybean.agronomy.wisc.edu.netbios-ns > typhoon.mynet.org.netbios-ns:NBT UDP PACKET(137): QUERY; REQUEST; UNICAST

18:12:00.469611 < soybean.agronomy.wisc.edu.netbios-ns > cd_pc.mynet.org.netbios-ns:NBT UDP PACKET(137): QUERY; REQUEST; UNICAST

 # And a SYN – RST couple to ftp on a Unix box
18:12:03.825178 < soybean.agronomy.wisc.edu.3499 > sunder.mynet.org.ftp: S 2020131264:2020131264(0) win 8192 <mss 1460> (DF) (ttl 116, id 13986)
18:12:03.825211 < sunder.mynet.org.ftp > soybean.agronomy.wisc.edu.3499: R 0:0(0) ack 2020131265 win 0 (DF) (ttl 116, id 46901)

. . . . 155 qualitatively similar records . . . .

#---- grep soybean.agronomy.wisc.edu /etc/syslog ----------

All these Unix boxes are tcp-wrapped
Jun 27 18:12:39 octopus.mynet.org ftpd[205099]: refused connect from soybean.agronomy.wisc.edu
Jun 27 18:12:39 ochre.mynet.org ftpd[662135]: refused connect from soybean.agronomy.wisc.edu
Jun 27 18:12:39 octal.mynet.org ftpd[298552]: refused connect from soybean.agronomy.wisc.edu
Jun 27 18:12:39 ocarina.mynet.org ftpd[101948]: refused connect from soybean.agronomy.wisc.edu

 # spite is an anonymous ftp server
Jun 27 18:12:54 spite.mynet.org in.ftpd[10479]: connect from soybean.agronomy.wisc.edu

. . . . 85 qualitatively similar records . . . .

#----- egrep '10479|soybean.agronomy.wisc.edu' /var/adm/messages ---

Jun 27 18:12:54 spite ftpd[10479]: connection from soybean.agronomy.wisc.edu at Wed Jun 27 18:12:54 2001
Jun 27 18:12:54 spite ftpd[10479]: <--- 220
Jun 27 18:12:54 spite ftpd[10479]: spite NIH LCP-NMR FTP Server (Version 5.60N) ready.
Jun 27 18:12:54 spite ftpd[10479]: command:
Jun 27 18:12:54 spite ftpd[10479]: <--- 500
Jun 27 18:12:54 spite ftpd[10479]: '': command not understood.
Jun 27 18:12:54 spite ftpd[10479]: command:
Jun 27 18:12:54 spite ftpd[10479]: <--- 500

```
Jun 27 18:12:54 spite ftpd[10479]: '': command not understood.
Jun 27 18:12:54 spite ftpd[10479]: command:
Jun 27 18:12:54 spite ftpd[10479]: <--- 500
Jun 27 18:12:54 spite ftpd[10479]: '': command not understood.
Jun 27 18:12:54 spite ftpd[10479]: <--- 221
Jun 27 18:12:54 spite ftpd[10479]: You could at least say goodbye.
```

1) Source of Trace:
My Laboratory

2) Detect was Generated by:
Site maintains 6 IDS sensors running tcpdump 24/7.

```
% tcpdump –w dumpFile_24hour
% snort –r dumpFile_24hour -c /etc/snort/snort.conf
```

At midnight, on a designated analysis workstation, process data using several shell, sed, awk, and perl scripts.
Gather all outside IP addresses and suspicious inside IP address of interest.
Obtain tcpdump data in separate files, for each "IP address of interest".

```
% tcpdump –r dump_24hour -vv "dst $outsider or src $outsider" \
        > tcpdump_IPaddress
% tcpdump -r dumpFile_24hour –vv -x \
        "dst $outsider or src $outsider" > tcpdump_hex_IPaddress
```

Snort Rule that generated this detect:
preprocessor portscan:

3) Probability that Source Address was Spoofed:
None
Reason: This was a UDP netbios scan, concurrent with TCP ftp connections as they were found. The hostile wanted netbios information, requiring that the IP address not be spoofed. And he wanted to penetrate anonymous ftp servers as he found them, necessitating a 3-way TCP hand shake. So lphm.phm.utoronto.ca and soybean.agronomy.wisc.edu are certainly the bad guys. (And, I always thought those upper midwest and Canadian guys were such honest, above board, and straight shooters. Another myth. . . down the tubes!)

4) Description of Attack:

From:	128.100.70.4	Ports 137 and Ephemeral ports
From:	128.104.35.50	Ports 137 and Ephemeral ports
To:	mynet.org	Ports 137 (netbios name service) and 21 (ftp)
Average Interval:	1.4 seconds	
Protocol:	UDP and TCP (netbios-ns:NBT UDP PACKET(137): QUERY; REQUEST; UNICAST)	

ID Numbers: Increment normally
Seq. Numbers: Increment normally
Ack. Numbers: Increment normally
TOS: [tos 0x10] "Minimize Delay"...unusual

These incidents were stimuli followed by responses.
This incident involved no crafted packets rather it used "Normal TCP/IP" traffic.
This incident was moderately fast (> 42 and 44 hits/minute).
This incident involved a moderate volume of traffic. It encompassed two entire subnets, 4 buildings, and several other segments of the entire organization.
This incident was a reconnaissance and exploitation of vulnerability as discovered.
The target OS for this incident appears to be writable anonymous ftp servers, and perhaps un-patched ftp servers.
The source OS for this incident is likely a Unix box.

5) Attack Mechanism:
This incident differs from the more or less daily attempts of hostiles to scan mynet.org for ftp sites, in which the hostile logs onto ftp servers, gathers banners, attempts to deposit files, or attempts a buffer overflow. The question is, why would someone probe 137 ports (netbios name server) associated with windows operating systems interspersed with ftp connections attempts, a service that can be installed on windows, but is usually associated with Unix boxes? I have been doing tcpdump/snort IDS for less than a month, but I have been doing tcpwrappers/central syslog server IDS for several years. I have hundreds of ftp attempts on my ftp servers. I have never seen this:

```
Jun 27 18:12:54 spite ftpd[10479]: '': command not understood.
```

Now, in one day, I have two detects with this signature. When I log onto my ftp servers from several sites as several different logins and experiment I can generate this error message by executing a " ^ G" or some other such nonsense character. I do not have access to a windows ftp server, but the windows ftp client will not generate this error message in the course of normal traffic.

Clearly, this is a UDP/137 scan followed by a TCP/21 attempt. In most every case the TCP/21 attempt follows the discovery of a candidate computer following the UDP/137 packet. The Unix boxes do not generally listen to port 137. I'm sure that the hostile responds to the "ICMP Destination Unreachable Port" packet returned by the Unix box in response. The fact that he uses a UDP/137 for discovery, suggests to me that he is looking for ftp servers on windows NT/2000 boxes (perhaps some third party ftp server on windows 9x). Presumably he wants access to windows ftp servers for the same reasons that motivate hostile Unix based ftp activity (file deposition and trojan or virus deposition). Remember, that windows ftp servers do not run in a chrooted environment, like well configured Unix ftp servers do, so passwd and sensitive file acquisition will be easier. When he finds a windows ftp server, I assume that the signature I see in the syslog files (': command not understood) may reflect activity that affords him some success.

However when he hits the much more common Unix ftp server, the detect generates the "noise" seen in my detect.

The theory, that this is a search for windows ftp servers has a counter argument. If he has no interest in Unix ftp servers, why doesn't he sort the UDP/137 responses from windows boxes from the "ICMP Destination Unreachable Port" messages, and target only what he wants, thereby leaving fewer foot prints? It is also possible that he was targeting linux boxes running SMB.

6) Correlations:
Hacking Exposed, Scambray, McClure, and Kurtz, page 45, Second Edition, Osborne/McGraw Hill (2001)

As indicated in "Detect 2", there are many vulnerabilities and CVEs associated with ftp. See Detect 2 for a list.

7) Evidence of Active Targeting:
This incident evidenced active targeting.
The attack was not "aimed" at specific workstation.
It was aimed at a specific series of networks.
And it was aimed at a particular service or ports (21, ftp and 137, netbios name service).

8) Severity:
Severity = (Criticality + Lethality) – ($CounterMeasures_{System}$ + $CounterMeasures_{Network}$)
Severity = (4 + 4) - (5 + 1)
Severity = 2

9) Defensive Recommendation:
Damages: No damage done.

This incident did penetrate the organizational firewall. On the other hand we maintain anonymous ftp sites for appropriate and necessary scientific communication between our scientists and the greater scientific community. This incident was automatically locked out of the ftp server within a few seconds of his first ftp attempt. And, so he was locked out of the main laboratory ftp server, portal, before he reached it. He was not, however, locked out of the second specialized ftp server, spite. I remain somewhat concerned that these two detects present a new signature (port 21 and port 137), however, I take comfort in that the 2 modes of intrusion in this incident seem to be working at cross purposes (platforms) with one another.

It is not reasonable to block this port at the firewall. This service needs to be placed in a DMZ outside of a laboratory firewall.

10) Multiple Choice Test Question:

What features of this detect illustrate the shortcomings of just relying syslogs/messages as the laboratory IDS.

a) The syslog server will record the failed and refused ftp attempts, but it will completely miss the port UDP/137 packets.
b) The "command not understood" in the messages logs, while odd, would perhaps not trigger the analysts concern without the other odd features in this detect. (After all, ftp has lots of error messages; I am still discovering them.)
c) The syslogs for this detect, in that they only reflect Unix traffic, are essentially identical to that for a "Looking for a Misconfigured FTP Server" detect as illustrated in detect 2.
d) All the above.

Detect 4: DNS SYN Scan

Summary:

From:	168.167.14.160	Ephemeral ports
To:	mynet.org	Port 53 (dns)
Notice:	SYN scan to port 53	
	No evidence of packet crafting	

[root@liar Process]# geektools 168.167.14.160
[whois.geektools.com]
Query: 168.167.14.160
Registry: whois.arin.net
Results:
University of Botswana (NET-BOTSNET)
Private Bag 0022,
Gaborone, Botswana
ZA

nslookup 168.167.14.160

160.14.167.168.in-addr.arpa name = mashadi.ub.bw.

Observed Traffic:
I have been doing tcpdump/snort IDS more or less properly since June 18 of this year (about 22 days). Already, I have had DNS scans from these addresses (some of them are repeat offenders):

128.104.35.50
mynet.100.14 This needs reporting to myorg.org
168.167.14.160
193.140.46.43
myorg.76.141 This needs reporting to myorg.org
198.87.182.135
200.4.128.218
207.8.203.106
208.221.194.5
210.62.176.151
211.152.32.14
211.186.87.114
61.218.125.26
62.140.64.160
62.168.94.10
66.1.254.166

The good news is that I do not run DNS service on my network, rather I rely on the 3 organization wide DNS servers. The bad news is that I really rely on those DNS servers. My colleagues, doing IDS at the organizational level suggest that they sometimes see this many incidents directed at the dns servers per day. So this, while not directly affecting me, has potentially serious impact on my network integrity. All these incidents look like this:

#---- grep 68.167.14.160 portscan.log ----------

```
Jun 22 09:28:48 168.167.14.160:1067 -> mynet.4.26:53 SYN ******S*
Jun 22 09:28:48 168.167.14.160:1091 -> mynet.4.50:53 SYN ******S*
Jun 22 09:28:48 168.167.14.160:1092 -> mynet.4.51:53 SYN ******S*
Jun 22 09:28:49 168.167.14.160:1163 -> mynet.4.122:53 SYN ******S*
Jun 22 09:28:51 168.167.14.160:1093 -> mynet.4.52:53 SYN ******S*
Jun 22 09:28:52 168.167.14.160:1222 -> mynet.4.181:53 SYN ******S*
Jun 22 09:28:52 168.167.14.160:1276 -> mynet.4.235:53 SYN ******S*
Jun 22 09:28:53 168.167.14.160:1353 -> mynet.5.57:53 SYN ******S*
Jun 22 09:28:54 168.167.14.160:1373 -> mynet.5.77:53 SYN ******S*

   . . . . 38 qualitatively similar records . . . .
   . . . . 485 detects from this and other IP addresses . . . .
```

#----- tcpdump from src 168.167.14.160 ----------

```
09:28:52.523598 < mashadi.ub.bw.1276 > sunder.mynet.org.domain: S
2762199139:2762199139(0) win 32120 <mss 1460,sackOK,timestamp
190947009[|tcp]> (DF) (ttl 48, id 6134)
09:28:52.523657 < sunder.mynet.org.domain > mashadi.ub.bw.1276: R 0:0(0) ack 2762199140
win 0 (DF) (ttl 48, id 63917)

09:29:42.687205 < mashadi.ub.bw.1201 > spinet.mynet.org.domain: S
2764258581:2764258581(0) win 32120 <mss 1460,sackOK,timestamp
190947009[|tcp]> (DF) (ttl 48, id 6059)
09:29:42.687508 < spinet.mynet.org.domain > mashadi.ub.bw.1201: R 0:0(0) ack 2764258582
win 0 (DF) (ttl 48, id 26211)

   . . . . 155 qualitatively similar records . . . .
```

1) Source of Trace:
My Laboratory

2) Detect was Generated by:
Site maintains 6 IDS sensors running tcpdump 24/7.

```
% tcpdump –w dumpFile_24hour
% snort –r dumpFile_24hour -c /etc/snort/snort.conf
```

At midnight, on a designated analysis workstation, process data using several shell, sed, awk, and perl scripts.
Gather all outside IP addresses and suspicious inside IP address of interest.
Obtain tcpdump data in separate files, for each “IP address of interest”.

```
% tcpdump –r dump_24hour -vv "dst $outsider or src $outsider" \
        > tcpdump_IPaddress
% tcpdump -r dumpFile_24hour –vv -x \
        "dst $outsider or src $outsider" > tcpdump_hex_IPaddress
```

Snort Rule that generated this detect:
preprocessor portscan:

3) Probability that Source Address was Spoofed:
Unlikely
Reason: You can imagine a port 53 scan as some kind of DOS and thus a candidate for spoofing, but more likely, the hostile is looking for a bind version, or for a misconfigured dns server from which he can obtain a zone transfer thereby gathering information for future attacks on other elements of the network. This is most likely a reconnaissance probe looking for any workstation that will respond to a port 53 packet. After he finds one he will be back with other tools, and at that time he may or may not spoof his address. This is almost surely the hostiles current IP address.

4) Description of Attack:

From:	168.167.14.160	Ephemeral ports
To:	mynet.org	Port 53 (dns)

This incident was a stimulus event.
This incident involved only "Normal TCP/IP" traffic.
This incident was moderately fast (> 22 hits/minute).
This incident involved a substantial volume of traffic. It encompassed two subnets, 4 buildings, and other segments of the entire institution.
This incident was a reconnaissance scan.
The target OS for this incident appears to be Unix DNS servers.
The source OS for this incident is unknown.

Average Interval:	2.7 seconds
Protocol:	TCP
ID Numbers:	Increment normally
Seq. Numbers:	Increment normally
Ack. Numbers:	Increment normally

5) Attack Mechanism:
This is syn scan of port 53 (dns) on mynet.org. Had one of my workstations, responded to a port 53 SYN packet, by establishing a 3 way handshake, it would have likely been followed perhaps first by an nmap like tool to perform an operating system footprint characterization, and certainly by bind version requests, zone transfers, or even buffer overflow attempts. Perhaps he wants to masquerade as the dns server for our organization.

6) Correlations:
There are several vulnerabilities associated with dns servers (http://cve.mitre.org/cve/).

CVE-1999-0010
Denial of Service vulnerability in BIND 8 Releases via maliciously formatted DNS

messages.
CVE-1999-0024
DNS cache poisoning via BIND, by predictable query IDs.
CVE-1999-0048
Talkd, when given corrupt DNS information, can be used to execute arbitrary commands with root privileges.
CVE-1999-0101
Buffer overflow in AIX and Solaris "gethostbyname" library call allows root access through corrupt DNS host names.
CVE-1999-0184
When compiled with the -DALLOW_UPDATES option, bind allows dynamic updates to the DNS server, allowing for
malicious modification of DNS records.
CVE-1999-0223
Solaris syslogd crashes when receiving a message from a host that doesn't have an inverse DNS entry.
CVE-1999-0274
Denial of service in Windows NT DNS servers through malicious packet which contains a response to a query that wasn't made.
CVE-1999-0275
Denial of service in Windows NT DNS servers by flooding port 53 with too many characters.
CVE-1999-0299
Buffer overflow in FreeBSD lpd through long DNS hostnames.
CVE-1999-0745
Buffer overflow in Source Code Browser Program Database Name Server Daemon (pdnsd) for the IBM AIX C Set ++ compiler.
CVE-2000-0020
DNS PRO allows remote attackers to conduct a denial of service via a large number of connections.
CVE-2000-0335
The resolver in glibc 2.1.3 uses predictable IDs, which allows a local attacker to spoof DNS query results.
CVE-2000-0405
Buffer overflow in L0pht AntiSniff allows remote attackers to execute arbitrary commands via a malformed DNS response packet.
CVE-2000-0536
xinetd 2.1.8.x does not properly restrict connections if hostnames are used for access control and the connecting host does not have a reverse DNS entry.
CVE-2001-0050
Buffer overflow in BitchX IRC client allows remote attackers to cause a denial of service and possibly execute arbitrary commands via an IP address that resolves to a long DNS hostname or domain name.

7) Evidence of Active Targeting:

This incident did not evidence Active Targeting of a particular workstation. It did, however, target the subnet *.mynet.org. It was a TCP scan indiscriminately looking for vulnerable dns servers.

8) Severity:
Severity = (Criticality + Lethality) – ($CounterMeasures_{System}$ + $CounterMeasures_{Network}$)

On one hand:
Severity = (1 + 1) - (5 + 2)
Severity = -5
I do not have any dns servers (Criticality = 1), and attack is unlikely to succeed (Lethality = 1).

On the other hand:
In the computer security business:

"*We shall hang together, or surely we shall hang separately*"
…Benjamin Franklin.

I do not have dns servers, but I certainly rely on them from the organizational level. If they were compromised, it could severely and adversely affect my capacity to run my network. So perhaps, I should amend this to

Severity = (5 + 5) - (5 + 2)
Severity = 7

The correct answer must lie between these extremes, and it clearly depends on where you stand.

9) Defensive Recommendation:
DNS vulnerability is a "Top-Ten Most Critical Security Vulnerabilities", in fact it is number 1. http://www.sans.org/topten.htm

This was a scan. I have no dns servers; so, no harm no foul. But, I should notify the organizational IDS team, to alert them of the existence of dns probes inside the firewall. (It will be like preaching to the choir, as they will surely know of this probe and the 16 others that I observed over the last 22 days. Those guys are exquisitely tuned into dns insecurities and run a very tight, modern, and well patched, and chrooted dns service in an insecure academic environment.) I should tell the renegade scientist in my lab who runs his own dns server for that one computer only, that this activity is going on. I should request again (actually I should just get in line to request) that port 53 be blocked at the organization's porous firewall for all destinations except for the organizational dns servers. Certainly, I should continue to monitor for this kind of activity, and keep the organization apprised of any findings.

10) Multiple Choice Test Question:

Consider this detect:

```
09:28:52.523598 < mashadi.ub.bw.1276 > sunder.mynet.org.domain: S
2762199139:2762199139(0) win 32120 <mss 1460,sackOK,timestamp
190947009[|tcp]> (DF) (ttl 48, id 6134)
09:28:52.523657 < sunder.mynet.org.domain > mashadi.ub.bw.1276: R 0:0(0) ack 2762199140
win 0 (DF) (ttl 48, id 63917)
```

Which of the following statements are true:

a) This is an UDP scan for dns servers.
b) This is an ICMP echo request scan for dns serves.
c) The workstation sunder is running a dns server.
d) The fact that the ttl in both packets is the same is indicative of a spoofed source address.
e) None of the above.

Detect 5: UDP scan from a Possibly Trojaned Computer

Summary:

From:	myorg.76.141 Ports 1032 and 1035
From:	myorg.76.166 Ports 1032
To:	mynet.org Port 38293
Notice:	This is high port to high port communication (suspicious)
	UDP port scans of sections of my PC network
	There is a story here concerning Norton Corporate Antivirus Servers
	The TTLs are wrong for on campus traffic?
	I will conclude that this is likely a false positive.

This detect looks like high port number communication from a trojaned computer within the organization. As such, it is hostile, and suggests the existence of a compromised computer within the organization. If true, it requires immediate investigation.

It is also possible, that this detect is somewhat normal traffic associated with a **"Norton Corporate Anti-Virus Signature Update Server".** The evidence for this is that the ports observed in this detect are those associated with this service. At best, it is only somewhat normal, because if some other net within the organization is running this service, it should under **no** circumstances be probing the computers on *.mynet.org (which does not use this service). At very least I need to have a "discussion" with the owner of commons2.oer.od.myorg.org.

```
[root@liar Process]# nslookup myorg.76.141

myorg.77.198.in-addr.arpa      name = commons2.oer.od.myorg.org.
```

```
[root@liar Process]# nslookup myorg.76.166

** server can't find 166.76.myorg.in-addr.arpa.: NXDOMAIN
```

Observed Traffic:
This happens most days, but not every day? And, it sometimes happens twice or three times a day. There are roughly 24 hits per episode.

In the beginning, it happened at irregular times. The time interval between probes varied from 16+ hours to 108+ hours. Probes do not start at odd hours but at the same minute. Although, 2 start on the same hour and within 5 seconds of one another. So it is not some kind of cron job (Or some windows equivalent to cron.) Later, the pattern became more regular at 3 times a day, but again not every day, and not always 3 times a day.

Jun 20	02:28:28
Jun 21	10:26:05
Jun 22	12:47:53
Jun 23	04:48:44
Skipped 2 days	
Jun 26	04:51:21
Skipped 4 days	
Jul 1	16:01:04
Skipped 1 day	
Jul 3	08:01:42
Jul 4	08:00:30
Jul 5	00:02:40
Jul 6	00:01:45
Skipped 1 day	
Jul 8	00:02:16
Jul 9	00:01:14

And, then 3 days later, they all of a sudden become more regular

Jul12	00:00
	08:00
	16:01
Jul 13	00:01
	07:59
	16:07
Jul 14	00:01
	08:02
	16:01
Jul 15	00:01
	08:01
	16:01

Jul 16	00:00
	07:59
	16:01
July 21	08:02

All these detected scans were directed against PCs. There are no Unix IP addresses in the data. The same IP addresses are not scanned in each episode. Three IP addresses are scanned once during each of the first twelve episodes. There are 3 IP addresses that were scanned only one time during the first twelve episodes. On average any particular IP address was scanned during only 4 to 5 of the first 12 episodes.

This is **not** a case in which a UDP scan finds PCs that just happen to be turned on. Three PCs that are always on, 24/7, have never been probed. A fourth PC also on for 24/7 was only probed only twice.

This is **not** a case in which the "Norton Corporate Anti-Virus Signature Update Server" somehow knows which of the very few PCs in mynet.org just happen to be running the "Norton Anti-virus Client Package" and then performs a signature update on just those PCs. That is because most of the computers that were targeted by myorg.76.166 do not run Norton anti virus client. The few that I found that were running Norton Clients certainly did **not** choose to be updated from a corporate server outside of mynet.org. (A service, for which I am quite sure some one pays real money.)

The "rules", some of them informal, of this organization, are that one subnet within the organization does not scan another subnet without prior notification and permission.

This is thus a scan, performed at random times, sometimes at 04:00 on the morning, against random PC IP addresses. It does not feel like a "routine scan" from some kind of regular service (Norton Corporate Anti-Virus Signature Update Server for example), that has spilled over from one subnet within the Organization to my subnet in the organization.

#---- grep myorg.76. portscan.log ----------

```
Jun 20 02:28:28 myorg.76.166:1032 -> mynet.5.194:38293 UDP
Jun 20 02:28:28 myorg.76.166:1032 -> mynet.5.223:38293 UDP
Jun 20 02:28:28 myorg.76.166:1032 -> mynet.5.233:38293 UDP
Jun 20 02:28:28 myorg.76.166:1032 -> mynet.5.80:38293 UDP
   . . . . more records on this date . . . .

Jun 21 10:26:05 myorg.76.166:1032 -> mynet.4.96:38293 UDP
Jun 21 10:26:05 myorg.76.166:1032 -> mynet.5.167:38293 UDP
Jun 21 10:26:05 myorg.76.166:1032 -> mynet.5.178:38293 UDP
Jun 21 10:26:05 myorg.76.166:1032 -> mynet.5.73:38293 UDP
   . . . . more records on this date . . . .
```

Jun 22 12:47:53 myorg.76.141:1032 -> mynet.5.227:38293 UDP
Jun 22 12:47:53 myorg.76.141:1032 -> mynet.5.72:38293 UDP
Jun 22 12:47:53 myorg.76.141:1032 -> mynet.5.75:38293 UDP
Jun 22 12:47:55 myorg.76.141:1032 -> mynet.4.110:38293 UDP
. . . . more records on this date . . . .

Jul 1 16:01:04 myorg.76.141:1035 -> mynet.5.167:38293 UDP
Jul 1 16:01:04 myorg.76.141:1035 -> mynet.5.178:38293 UDP
Jul 1 16:01:05 myorg.76.141:1035 -> mynet.4.96:38293 UDP
Jul 1 16:01:05 myorg.76.141:1035 -> mynet.5.166:38293 UDP
. . . . more records on this date . . . .

Jul 9 00:01:14 myorg.76.141:1035 -> mynet.5.56:38293 UDP
Jul 9 00:01:15 myorg.76.141:1035 -> mynet.5.64:38293 UDP
Jul 9 00:01:15 myorg.76.141:1035 -> mynet.5.75:38293 UDP
Jul 9 00:01:16 myorg.76.141:1035 -> mynet.5.148:38293 UDP
. . . . more records on this date . . . .
. . . . 773 qualitatively similar records . . . .

#---- tcpdump from src myorg.76.141 ----------

12:47:42.925982 < commons2.oer.od.myorg.org.1032 > angela1.mynet.org.38293: udp 16 (ttl 28, id 42867)
12:47:43.258966 < commons2.oer.od.myorg.org.1032 > caipc.mynet.org.38293: udp 16 (ttl 28, id 56691)
12:47:50.811514 < commons2.oer.od.myorg.org.1032 > gwc_pc9.mynet.org.38293: udp 16 (ttl 28, id 27252)
12:47:53.243844 < commons2.oer.od.myorg.org.1032 > iwlpc5.mynet.org.38293: udp 16 (ttl 28, id 35700)
12:47:53.349833 < commons2.oer.od.myorg.org.1032 > joanpc.mynet.org.38293: udp 16 (ttl 28, id 40820)
12:47:53.498863 < commons2.oer.od.myorg.org.1032 > lapidus.mynet.org.38293: udp 16 (ttl 28, id 48756)
12:47:55.836395 < commons2.oer.od.myorg.org.1032 > shuko_pc1.mynet.org.38293: udp 16 (ttl 28, id 56692)
12:47:56.656340 < commons2.oer.od.myorg.org.1032 > olgica.mynet.org.38293: udp 16 (ttl 28, id 25205)

#---- tcpdump from src myorg.76.166

02:28:28.416495 < myorg.76.166.1032 > dgarrett3.mynet.org.38293: udp 16 (ttl 28, id 10118)
02:28:28.440748 < myorg.76.166.1032 > dtb_pc.mynet.org.38293: udp 16 (ttl 28, id 11654)
02:28:28.463556 < myorg.76.166.1032 > eh_pcWin.mynet.org.38293: udp 16 (ttl 28, id 12678)
02:28:28.521403 < myorg.76.166.1032 > femto2.mynet.org.38293: udp 16 (ttl 28, id 15494)
02:28:30.889747 < myorg.76.166.1032 > gwcpc2.mynet.org.38293: udp 16 (ttl 28, id 23942)
02:28:30.930889 < myorg.76.166.1032 > hjhpc.mynet.org.38293: udp 16 (ttl 28, id 25990)

1) Source of Trace:

My Laboratory

2) Detect was Generated by:

Site maintains 6 IDS sensors running tcpdump 24/7.

```
% tcpdump –w dumpFile_24hour
% snort –r dumpFile_24hour -c /etc/snort/snort.conf
```

At midnight, on a designated analysis workstation, process data using several shell, sed, awk, and perl scripts.
Gather all outside IP addresses and suspicious inside IP address of interest.
Obtain tcpdump data in separate files, for each "IP address of interest".

```
% tcpdump –r dump_24hour -vv "dst $outsider or src $outsider" \
        > tcpdump_IPaddress
% tcpdump -r dumpFile_24hour –vv -x \
        "dst $outsider or src $outsider" > tcpdump_hex_IPaddress
```

Snort Rule that generated this detect:
preprocessor portscan:

3) Probability that Source Address was Spoofed:
Unable to Tell
Reason: By default, and by centralized purchasing authority, the Organization buys and uses the McAffee antivirus package all over campus (Network Associates). To individual users and laboratories the McAffee anti virus services are "free". All the windows mail exchange servers on campus use McAffee tools to purge virus infected attachments from all windows mail. It seems both foolish and unlikely that one entire subnet (incidentally, in the administration office) is using the Norton antivirus system, although anything is possible. Obviously, if this is a Norton Antivirus Corporate Server in another part of the organization, which is running an improperly configured server, then this is not a spoofed IP address.

But if this is a trojan mimicking the Norton ports, then all bets are off. In particular, the organization runs a very porous firewall. By default it accepts all, with certain ports turned off. The anti-spoofing filters that blocks external traffic arriving with internal IP addresses are just being installed this month.

This suggests an elegant way to hide hostile traffic. An outside hostile, from his own or a compromised computer, might spoof source IP addresses looking for trojans that listen to port 38293. Most alarming of all, there could be a kind of Norton-Trojan sending packets port 38293 from IP addresses with source addresses crafted as true "Norton Corporate Antivirus Server". Such traffic would be "trusted" by victims because it is seen to be from the local virus update servers. The point of such software would be to poison pre-existing viral signature files, or to install other trojans, like SubSeven, or even to install Windows based DDOS client tools for later use in DDOS attacks against third party victims. (I am not a hacker, nor have I ever been one. But, if I played one on television…I would sure give this some serious thought.)

Arguing for the hypothesis that this is the Norton Antivirus Corporate server, is the fact that there are repeat scans to the same subnet . After all, if you are "trolling for trojans", and you do not find any, or if you are installing mal-ware that you installed yesterday why do it again on the same net on the next day. It would increase the noise and elevate the probability of discovery. And, perhaps the odd timing of the Norton scans occurs because they are triggered by the presence of new anti-viral information obtained from Symantec, rather than on a regular "cron-style" basis?

The best evidence I have that this is a spoofed address, derived from an off campus hostile computer is outlined next.

In the "detects", the TTL, for data packets sourced from myorg.76.141 and myorg.76.166 for the entire month, is 28 steps in all cases. But packets requested from commons2.oer.od.myorg.org by me at mynet.org, possess a TTL of 124 steps remaining.

#--- ping myorg.76.141 Look at TTL in "icmp: echo reply"
09:56:43.950813 eth0 > liar.mynet.org > commons2.oer.od.myorg.org: icmp: echo request (DF) (ttl 64, id 0)
09:56:43.950813 eth0 < commons2.oer.od.myorg.org > liar.mynet.org: icmp: echo reply (DF) (**ttl 124**, id 61675)

09:56:44.950785 eth0 > liar.mynet.org > commons2.oer.od.myorg.org: icmp: echo request (DF) (ttl 64, id 0)
09:56:44.950785 eth0 < commons2.oer.od.myorg.org > liar.mynet.org: icmp: echo reply (DF) (**ttl 124**, id 61931)

#--- The same is true when attempting to form a 3-way TCP handshake
10:06:44.763826 eth0 > liar.mynet.org.33376 > commons2.oer.od.myorg.org.ftp: S 472245501:472245501(0) win 5840 <mss 1460,sackOK,timestamp 143966864 0,nop,wscale 0> (DF) (ttl 64, id 9837)
10:06:44.763826 eth0 < commons2.oer.od.myorg.org.ftp > liar.mynet.org.33376: R 0:0(0) ack 472245502 win 0 (**ttl 124**, id 47605)

By default, Windows NT sends TCP/IP packets with TTL set to 128 (http://www.map2.ethz.ch/ftp-probleme.htm). So, a TTL of 124 for an on campus NT box to communicate with my network is reasonable. But the "hostile" detect has a TTL of 28. Hmmm? I can think of only two possible explanations. The benign one is that Symantec designed the Server to send packets with TTLs of 32. Perhaps they wanted update packets to not travel far in the event that they escaped the local network? The less benign explanation is that the address commons2.oer.od.myorg.org is being spoofed by an off campus site that is over 100 hops away. I already know that the Organization firewall is not blocking packets with both source and destination IP addresses from on campus.

So there is insufficient information to determine if the address is spoofed. Of course the technical person in that network is not in the office for two weeks, and I cannot ask him. I have found out from third parties, that this IP address, while under the rubric of the organization, is in fact under the control of an off campus contract company. This

company does run "Norton Antivirus Corporate Server". So the likely hood is high that this activity represents a mis-configured viral update server.

There remain the issues of why they direct traffic at my network, how their target PCs are seemingly randomly selected. I cannot explain why the traffic is directed at PCs without Norton Antivirus Clients; and I cannot explain why the traffic is not directed at the same PCs on each and every scan. Finally, what is this IP address myorg.76.166, that is responsible for 2 of the scans. I have never been able to ping it, and it is not on the local dns. Is it a backup which is normally off? Do they have 2 servers? Is it a spoofed address?

I called Symantec (Norton Anti Virus). Eventually, I found a technical support person who was willing to confirm that the Norton Antivirus Corporate Server product does indeed communicate through destination port 38293. I was unable to clearly ascertain from the conversation whether the NAV server "pushes" its update information which could explain why my network sees the packets from a misconfigured server, or whether the clients must proactively "pull" the information. We are now undertaking an e-mail correspondence in which I am trying to obtain the value of the TTL parameter used by the product for UDP communication.

4) Description of Attack:

From:	myorg.76.141	Ports 1032 and 1035
From:	myorg.76.166	Ports 1032
To:	mynet.org	Port 38293

Average Interval:	4.6 seconds
Protocol:	UDP
ID Numbers:	Varies normally
TTL:	28

This incident was a stimulus.
It is unclear whether this incident involved crafted packets or indicated "Normal TCP/IP" traffic. Although, the simplest assumption is that someone is running a misconfigured Norton antivirus corporate server.
This incident was moderately fast (~ 13 hits/minute).
This incident involved a small volume of traffic. It encompassed PCs only on parts of 2 subnets, in a single building.
This incident was either a false positive, perhaps a reconnaissance, perhaps a search for trojans, or perhaps an attempt to install mal-ware on PCs that might accept them.
The target OS for this incident was windows 9x/NT/2000 running on a PC.
The source OS for this incident appears to be another 9x/NT/2000 running on a PC, probably NT/2000.

5) Attack Mechanism:

This was likely Norton Antivirus Corporate server activity from a misconfigured server.

6) Correlations:
I found two sights that offer some information about the Norton AntiVirus Corporate Edition".
www.sans.org/y2k/092300.htm
www.sans.org/y2k/032101.htm

"Norton AntiVirus Corporate Edition" uses port number 38293 for client-to-server communication Trust this issue if Norton Corporate Edition Anti-Virus is installed."
http://www.networkice.com/Advice/Intrusions/2003412/default.htm

7) Evidence of Active Targeting:
This incident certainly evidenced active targeting of *.mynet.org. In addition, because different PCs were targeted during different episodes, there is the likelihood that only certain IP addresses were probed during a particular episode. The detect exhibits "negative evidence" of active targeting in that only PCs were probed, the detect involves no Unix boxes whatsoever. (Unix boxes constitute the majority of the assets on my network.). If that is true, then specific PCs were targeted. I cannot think of a scenario in which the IP addresses that were probed sensibly correlates with those PCs that were powered on during the episode.

8) Severity:
Severity = (Criticality + Lethality) – ($CounterMeasures_{System}$ + $CounterMeasures_{Network}$)
Severity = (2 + 4) - (2 + 1)
Severity = 3

Comment: These are PCs (Criticality = 1). If this is a Norton Antivirus Server, then the lethality = 1 (0?). Of course if it is from an off campus site, and it is a "Norton-Trojan", then the lethality will be much higher. Even though I plead guilty to the occasional Unix users "anti-windows bigotry", this is likely more serious than the potential viral infection of a single windows box. If a virus penetrates the Organizations anti virus defenses, and if that virus does not come in as an e-mail attachment (that would be cleaned at the Microsoft Exchange mail servers), and if that virus is one of the modern ones that forwards itself to the entire mailing list (thousands of people in the Organization) then the possibility of serious Organization wide viral infection is very real. The first such virus to hit the Organization "I Love You", incapacitated the network mail system for several days. This escalates the Lethality to perhaps 4 or 5.

9) Defensive Recommendation:
Damages: None yet.
Firewall rules. Clearly this incident penetrated the firewall, either by stealth or by official installation and misconfiguration. I have been unable to track down the administrator for this network. Those I have tracked down have not understood my questions, passed me

on to others, or were out of the office till next week. And, I have not yet been able to gather all the pertinent information from Symantec. So, while the resolution of this detect is not quite finished for the GIAC submission date, I will peruse this to either my personal embarrassment, or to its technical conclusion.

10) Multiple Choice Test Question:

Which of these statements are true.

a) TTL values are the number of seconds that an IP packet will remain active before a router drops the packet.
b) All operating systems create packets with a TTL value of 128.
c) TTL values are variables that apply to TCP traffic only, the concept does not exist in UDP and ICMP traffic.
d) The TTL for a packet that comes from there.org to here.org, should usually be the same as the TTL for a packet that goes from here.org to there.org, as seen on the IDS installed at here.org.
e) None of the above.

Detect 6: Looking for Sub Seven

Summary:

From:	209.214.160.1	Ephemeral ports
From:	24.101.119.83	Ephemeral ports
From:	24.157.161.226	Ephemeral ports
To:	mynet.org	Port 1243
Notice:		

```
[root@liar Process]# geektools 209.214.160.1
[whois.geektools.com]
Query:    209.214.160.1
Registry:  whois.arin.net
Results:
BellSouth.net Inc. (NETBLK-BELLSNET-BLK4)
   301 Perimeter Center North,  Suite 400
   Atlanta, GA 30346
   US

[root@liar Process]# geektools 24.101.119.83
[whois.geektools.com]
Rogers@Home MTWH (NETBLK-ON-ROG-8-1MTWH-6) ON-ROG-8-1MTWH-6
            24.101.119.0 - 24.101.119.255

[root@liar Process]# arin NETBLK-ON-ROG-8-1MTWH-6
[whois.arin.net]
Rogers@Home MTWH (NETBLK-ON-ROG-8-1MTWH-6)
   1 Mount Pleasant Road
   Toronto, ON M4Y 2Y5
   CA

[root@liar Process]# geektools 24.157.161.226
[whois.geektools.com]
Rogers@Home Ktchnr (NETBLK-ON-ROG-2-3KTCHNR-2) ON-ROG-2-3KTCHNR-2
              24.157.161.128 - 24.157.161.255

Rogers@Home Ktchnr (NETBLK-ON-ROG-2-3KTCHNR-2)
   1 Mount Pleasant Road
   Toronto, ON M4Y 2Y5
   CA

[root@liar Process]# nslookup 209.214.160.1
1.160.214.209.in-addr.arpa  name = host-209-214-160-1.rdu.bellsouth.net.

[root@liar Process]# nslookup 24.101.119.83
83.119.101.24.in-addr.arpa   name = 24.101.119.83.on.wave.home.com.

[root@liar Process]# nslookup 24.157.161.226
226.161.157.24.in-addr.arpa  name = cr166156-a.ktchnr1.on.wave.home.com.
```

Sarcastic Comment: I may only have been doing tcpdump/snort IDS for a month, but I

have been focused on Unix computer security and have monitored syslogs/messages since 1994, after my first root compromise. I must say, you could probably build a career on following the hostile antics from *.home.com and *.wanadoo.fr.

Observed Traffic:

On June 23, there were 61 detects of packets looking for computers compromised with SubSeven. They derived from these IP addresses:

Count	Port
43	209.214.160.1
10	24.157.161.226
8	24.101.119.83

These happened between: 06/23-23:17:19.829544 -> 06/23-23:18:45.980296 (less than 2 minutes). This, and the fact that two of the three IP addresses are from *.on.wave.home.com suggests that all 3 incidents were controlled by the same hostile.

#---- One complete episode of SubSeven search from 24.157.161.226 –
 # These are all directed against windows9x/NT/2000 boxes.

[**] Possible SubSeven access [**]
06/23-23:17:19.829544 209.214.160.1:1775 -> mynet.5.56:1243
TCP TTL:108 TOS:0x0 ID:23100 IpLen:20 DgmLen:48 DF
******S* Seq: 0x355D3D Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK

[**] Possible SubSeven access [**]
06/23-23:17:19.830030 209.214.160.1:1776 -> mynet.5.57:1243
TCP TTL:108 TOS:0x0 ID:23356 IpLen:20 DgmLen:48 DF
******S* Seq: 0x355D3E Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK

[**] Possible SubSeven access [**]
06/23-23:17:30.399180 209.214.160.1:1790 -> mynet.5.72:1243
TCP TTL:108 TOS:0x0 ID:46652 IpLen:20 DgmLen:48 DF
******S* Seq: 0x358AC2 Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK

[**] Possible SubSeven access [**]
06/23-23:17:30.412061 209.214.160.1:1793 -> mynet.5.75:1243
TCP TTL:108 TOS:0x0 ID:47420 IpLen:20 DgmLen:48 DF
******S* Seq: 0x358AC5 Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK

[**] Possible SubSeven access [**]
06/23-23:17:30.412582 209.214.160.1:1792 -> mynet.5.74:1243
TCP TTL:108 TOS:0x0 ID:47164 IpLen:20 DgmLen:48 DF
******S* Seq: 0x358AC4 Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK

[**] Possible SubSeven access [**]
06/23-23:17:30.439440 209.214.160.1:1798 -> mynet.5.80:1243

TCP TTL:108 TOS:0x0 ID:48700 IpLen:20 DgmLen:48 DF
******S* Seq: 0x358AC9 Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK

[**] Possible SubSeven access [**]
06/23-23:17:42.319263 209.214.160.1:1810 -> mynet.5.93:1243
TCP TTL:108 TOS:0x0 ID:3901 IpLen:20 DgmLen:48 DF
******S* Seq: 0x35B7A2 Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK

[**] Possible SubSeven access [**]
06/23-23:17:44.750108 209.214.160.1:1817 -> mynet.5.100:1243
TCP TTL:108 TOS:0x0 ID:12093 IpLen:20 DgmLen:48 DF
******S* Seq: 0x35B7A9 Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK

[**] Possible SubSeven access [**]
06/23-23:17:53.760742 209.214.160.1:1837 -> mynet.5.121:1243
TCP TTL:108 TOS:0x0 ID:25149 IpLen:20 DgmLen:48 DF
******S* Seq: 0x35E48D Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK

[**] Possible SubSeven access [**]
06/23-23:17:53.811172 209.214.160.1:1846 -> mynet.5.130:1243
TCP TTL:108 TOS:0x0 ID:27453 IpLen:20 DgmLen:48 DF
******S* Seq: 0x35E496 Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK

#---- Another episode of SubSeven search from 24.157.161.226 ---
 # This is directed against a Solaris NIS server.

[**] Possible SubSeven access [**]
06/23-23:17:42.401880 24.157.161.226:2159 -> mynet.4.235:1243
TCP TTL:110 TOS:0x0 ID:47319 IpLen:20 DgmLen:48 DF
******S* Seq: 0x77DD55E Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

#---- tcpdump from src 209.214.160.1 -------------------------
 # At present, the IDS does not see the RST packets returned by PCs

```
23:17:19.829544 < host-209-214-160-1.rdu.bellsouth.net.1775 >
mynet.5.56.1243: S 3497277:3497277(0) win 8192 <mss 536,nop,nop,sack
OK> (DF) (ttl 108, id 23100)

23:17:19.830030 < host-209-214-160-1.rdu.bellsouth.net.1776 >
rolfpc2.mynet.org.1243: S 3497278:3497278(0) win 8192 <mss 536,nop
,nop,sackOK> (DF) (ttl 108, id 23356)

23:17:30.399180 < host-209-214-160-1.rdu.bellsouth.net.1790 >
iwlpc5.mynet.org.1243: S 3508930:3508930(0) win 8192 <mss 536,nop,
nop,sackOK> (DF) (ttl 108, id 46652)

23:17:30.412061 < host-209-214-160-1.rdu.bellsouth.net.1793 >
```

```
lapidus.mynet.org.1243: S 3508933:3508933(0) win 8192 <mss 536,nop
,nop,sackOK> (DF) (ttl 108, id 47420)

23:17:30.412582 < host-209-214-160-1.rdu.bellsouth.net.1792 >
tras.mynet.org.1243: S 3508932:3508932(0) win 8192 <mss 536,nop,no
p,sackOK> (DF) (ttl 108, id 47164)
```

. . . . 209 qualitatively similar records . . . .

#---- tcpdump from src 24.157.161.226 (SYN – RST pair) ----------

```
23:17:42.401880 < cr166156-a.ktchnr1.on.wave.home.com.2159 > sunder.mynet.org.1243: S
125687134:125687134(0) win 8192 <mss 14
60, nop,nop,sackOK> (DF) (ttl 110, id 47319)

23:17:42.401907 < sunder.mynet.org.1243 > cr166156-a.ktchnr1.on.wave.home.com.2159: R
0:0(0) ack 125687135 win 0 (DF) (ttl 110, id 9679)
```

1) Source of Trace:
My Laboratory

2) Detect was Generated by:
Site maintains 6 IDS sensors running tcpdump 24/7.

```
% tcpdump –w dumpFile_24hour
% snort –r dumpFile_24hour -c /etc/snort/snort.conf
```

At midnight, on a designated analysis workstation, process data using several shell, sed, awk, and perl scripts.
Gather all outside IP addresses and suspicious inside IP address of interest.
Obtain tcpdump data in separate files, for each "IP address of interest".

```
% tcpdump –r dump_24hour -vv "dst $outsider or src $outsider" \
        > tcpdump_IPaddress
% tcpdump -r dumpFile_24hour –vv -x \
        "dst $outsider or src $outsider" > tcpdump_hex_IPaddress
```

Snort Rule that generated this detect:
From the backdoor-lib:

```
alert tcp any any -> $HOME_NET 1243 (msg:"Possible SubSeven access"; flags: S;)
```

3) Probability that Source Address was Spoofed:
Very low
Reason: This was a scan looking for pre installed SubSeven trojans on mynet.org. In order to learn of the success of this probe, the presumably infected workstation must establish 3 way TCP communication. The source was not spoofed.

4) Description of Attack:

From:	209.214.160.1	Ephemeral ports
From:	24.101.119.83	Ephemeral ports
From:	24.157.161.226	Ephemeral ports
To:	mynet.org	Port 1243

Average Interval:	1.4 sec
Protocol:	TCP
ID Numbers:	Normal variation
Seq. Numbers:	Normal variation
Ack. Numbers:	Normal variation
TTL:	Unremarkable
Window Size:	Unremarkable

This incident was a stimulus event.
This incident involved only "Normal TCP/IP" traffic.
This incident fast scan (~ 42 hits/minute).
This incident involved a moderately small volume of traffic. It encompassed a single subnet in one building.
This incident was a reconnaissance probe. It would presumably have involved an immediate, or at least a near immediate exploitation of SubSeven, had it been discovered.
The target OS for this incident appears to be windows 9x/NT/2000.
The source OS for this incident is likely windows 9x/NT/2000

5) Attack Mechanism:

SubSeven is a Windows 9x/NT trojan. It originated in the Netherlands, was built by "mobman", and released in 1999. It is also known as Backdoor-G, Sub7, and Backdoor-G2. It is a client/server application. A windows system becomes compromised in any of the several ways that PCs acquire trojans, usually by inadvertent downloading from the web or installation by e-mail attachment execution. The server runs on the compromised host and controls programs on the compromised host. The server can be manipulated by clients on other computers through a TCP port. The port over which it communicates is configurable, but defaults to TCP/27374. It is often seen on ports 6711, 6776, 1243 (as in this detect), and 1999. Intruders often undertake scans of networks looking for compromised PCs running SubSeven.

6) Correlations:

There is a CAN number for trojans. (CANs are Canditates for inclusion on the CVE list)
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660

The SubSeven trojan has an "Incident Note" at CERT.
http://www.cert.org/incident_notes/IN-2001-07.html

And, according to CERT, there have been recent increases in SubSeven activity:
http://www.cert.org/current/current_activity.html

There have been several GIAC practicums with detects and analysis of SubSeven. http://www.sans.org/giactc/gcia.htm. c.f.

Robert Currie
Kevin Orkin
David Singer

And a NIPC advisories.
"New Scanning Activity (with W32-Leaves.worm) Exploiting SubSeven Victims "
http://www.nipc.gov/warnings/advisories/2001/01-014.htm,
"SubSeven DEFCON8 2.1 Backdoor" Trojan
http://www.nipc.gov/warnings/advisories/2000/00-056.htm

7) Evidence of Active Targeting:
This incident did evidence Active Targeting.
The attack was "aimed" at specific network, and it scanned most of that network.
Or attack was aimed only against port 1243 on windows OS.

8) Severity:
Severity = (Criticality + Lethality) – ($CounterMeasures_{System}$ + $CounterMeasures_{Network}$)
Severity = (1 + 3) - (4 + 1)
Severity = -1

Comment: Our organization supports centrally administered anti-virus software support from McAffee which will find and remove SubSeven. In principle all organizational computers routinely run updated versions this software. In practice this is more true than not.

9) Defensive Recommendation:
Damages: No SubSeven trojans were discovered buy this probe. Maintain due diligence over user PCs and insure their ongoing use of anti-virus programs.
It would seem prudent to block targeted ports 1243 at the porous organizational firewall. On the other hand SubSeven can be configured to use other ports, so this may not be very effective.

10) Multiple Choice Test Question:

Select the true statement:

a) SubSeven always communicates on port TCP/1243.
b) SubSeven can be used to manipulate Unix workstations.
c) The hostile package SubSeven is a bad thing. The commercial product PCanywhere is a good thing because…. On second thought, maybe it's not.
d) All of the above.

Detect 7: Possible WinGate Activity

Summary:

From: 202.39.9.109 Ports 1085, 2008, 2013, 1094, 1087
To: mynet.4.245 Ports 8080, 1085, 7000, 8000
To: mynet.4.52 Port 2008
Notice: It is not exclusively a WinGate (src Port 1080) search.
Snort discovered src port 1080, and
this is likely Wingate,
But, this looks at other high number ports.
He is probably for other trojaned sites.

```
[root@liar Process]# geektools 202.39.9.109
[whois.geektools.com]
Query:     202.39.9.109
Registry:  whois.apnic.net
Results:

% Rights restricted by copyright. See http://www.apnic.net/db/dbcopyright.html
% (whois5.apnic.net)

inetnum:     202.39.0.0 - 202.39.255.255
netname:     TWNIC-TW
descr:       Taiwan Network Information Center
descr:       4F-2, No. 9 Sec. 2, Roosevelt Rd.,
descr:       Taipei, Taiwan, 100
country:     TW

[root@liar Process]# nslookup 202.39.9.109
** server can't find 109.9.39.202.in-addr.arpa.: NXDOMAIN
```

Observed Traffic:

#---- snort alert entry --------------------------

```
[**] MISC-WinGate-8080-Attempt [**]
07/05-06:13:54.804912 202.39.9.109:1085 -> mynet.4.245:8080
TCP TTL:100 TOS:0x0 ID:62601 IpLen:20 DgmLen:48 DF
******S* Seq: 0xAEB54404  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
```

#---- snort log files ----------------------------

 #   First echo requests and replies -------------------------

```
[**] ICMP Echo Request [**]
07/05-06:13:34.582235 202.39.9.109 -> mynet.4.184
ICMP TTL:100 TOS:0x0 ID:42854 IpLen:20 DgmLen:64
Type:8  Code:0  ID:15368   Seq:0  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP Echo Request [**]
```

```
07/05-06:13:34.580428 202.39.9.109 -> mynet.4.184
ICMP TTL:100 TOS:0x0 ID:42854 IpLen:20 DgmLen:64
Type:8  Code:0  ID:15368   Seq:0  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP Echo Request [**]
07/05-06:13:35.136786 202.39.9.109 -> mynet.4.245
ICMP TTL:100 TOS:0x0 ID:43420 IpLen:20 DgmLen:64
Type:8  Code:0  ID:15368   Seq:0  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

#     Then WinGate 1080 attempts from different src ports --------
[**] MISC-WinGate-1080-Attempt [**]
07/05-06:14:10.340286 202.39.9.109:2008 -> mynet.4.245:1080
TCP TTL:100 TOS:0x0 ID:12153 IpLen:20 DgmLen:48 DF
******S* Seq: 0xB1BE32AA  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] MISC-WinGate-1080-Attempt [**]
07/05-06:14:11.950243 202.39.9.109:2008 -> mynet.4.245:1080
TCP TTL:100 TOS:0x0 ID:13792 IpLen:20 DgmLen:48 DF
******S* Seq: 0xB1BE32AA  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] MISC-WinGate-1080-Attempt [**]
07/05-06:14:10.421494 202.39.9.109:2013 -> mynet.4.52:1080
TCP TTL:100 TOS:0x0 ID:12165 IpLen:20 DgmLen:48 DF
******S* Seq: 0xB1C1A436  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] MISC-WinGate-1080-Attempt [**]
07/05-06:14:11.957616 202.39.9.109:2013 -> mynet.4.52:1080
TCP TTL:100 TOS:0x0 ID:13793 IpLen:20 DgmLen:48 DF
******S* Seq: 0xB1C1A436  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] MISC-WinGate-8080-Attempt [**]
07/05-06:13:54.804912 202.39.9.109:1085 -> mynet.4.245:8080
TCP TTL:100 TOS:0x0 ID:62601 IpLen:20 DgmLen:48 DF
******S* Seq: 0xAEB54404  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+


#---- tcpdump tells a more complicated story
#---- tcpdump from src 209.214.160.1 -------------------------

#     First some echo requests and replies
06:13:34.580428 < 202.39.9.109 > mynet.4.184: icmp: echo request
06:13:35.136786 < 202.39.9.109 > mynet.4.245: icmp: echo request
```

06:13:35.137082 < mynet.4.245 > 202.39.9.109: icmp: echo reply (DF)

 # Then the "WinGate" activity
06:13:54.804912 < 202.39.9.109.1085 > mynet.4.245.8080: S 2931115012:2931115012(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)

06:13:54.805185 < mynet.4.245.8080 > 202.39.9.109.1085: R 0:0(0) ack 2931115013 win 0 (DF)

06:14:10.340286 < 202.39.9.109.2008 > mynet.4.245.1080: S 2982032042:2982032042(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)

06:14:10.340553 < mynet.4.245.1080 > 202.39.9.109.2008: R 0:0(0) ack 2982032043 win 0 (DF)

 # Then WinGate request to port 1080 on a new workstation
 # (no IDS on workstation, so we do not see some RSTs)
06:14:10.419766 < 202.39.9.109.2013 > mynet.4.52.7000: S 2982257718:2982257718(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)

06:14:11.950512 < mynet.4.245.1080 > 202.39.9.109.2008: R 0:0(0) ack 1 win 0 (DF)

06:14:11.955828 < 202.39.9.109.2013 > mynet.4.52.1080: S 2982257718:2982257718(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)

 # But, packets to ports 7000 and 8000, not seen by snort –
06:13:54.831363 < 202.39.9.109.1094 > mynet.4.245.7000: S 2931551856:2931551856(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)

06:13:54.831648 < mynet.4.245.7000 > 202.39.9.109.1094: R 0:0(0) ack 2931551857 win 0 (DF)

06:13:54.851488 < 202.39.9.109.1087 > mynet.4.245.8000: S 2931219832:2931219832(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)

06:13:54.851779 < mynet.4.245.8000 > 202.39.9.109.1087: R 0:0(0) ack 2931219833 win 0 (DF)

1) Source of Trace:
My Laboratory

2) Detect was Generated by:
Site maintains 6 IDS sensors running tcpdump 24/7.

```
% tcpdump –w dumpFile_24hour
% snort –r dumpFile_24hour -c /etc/snort/snort.conf
```

At midnight, on a designated analysis workstation, process data using several shell, sed, awk, and perl scripts.
Gather all outside IP addresses and suspicious inside IP address of interest.
Obtain tcpdump data in separate files, for each "IP address of interest".

```
% tcpdump –r dump_24hour -vv "dst $outsider or src $outsider" \
        > tcpdump_IPaddress
% tcpdump -r dumpFile_24hour –vv -x \
        "dst $outsider or src $outsider" > tcpdump_hex_IPaddress
```

Snort Rule that generated this detect:
misc-lib:
```
alert tcp $EXTERNAL_NET !53 -> $HOME_NET 8080 (msg:"MISC-WinGate-8080-
Attempt";flags:S;)
```

And info.rules:
```
alert icmp !$HOME_NET any -> any any (msg:"ICMP Echo Request"; itype: 8; icode: 0;)
```

3) Probability that Source Address was Spoofed:
Unlikely
Reason: These packets were attempting to establish TCP 3 way handshake communication. These were a few packets directed to 2 workstations, presumably looking for a compromised port.

4) Description of Attack:

From:	202.39.9.109	Ports 1085, 2008, 2013, 1094, 1087
To:	mynet.4.245	Ports 8080, 1085, 7000, 8000
To:	mynet.4.52	Port 2008

Average Interval:	2.5 seconds
Protocol:	ICMP & TCP
ID Numbers:	Unremarkable
Seq. Numbers:	Unremarkable
TOS:	0x00
TTL:	100
Window Size:	Unremarkable

This incident was a stimulus event.
This incident involved only "Normal TCP/IP" traffic.
This incident was fast (~ 24 hits/minute).
This incident involved a small volume of traffic. It targeted 2 workstations only.
This incident was a reconnaissance
The target OS for this incident appears to be a Solaris workstations and a Network Appliance (raided disk farm).
The source OS for this incident is unknown.

5) Attack Mechanism:
Socks servers are multi-application proxy servers often found on firewalled networks. Hostiles can use socks and proxy servers to hide their addresses. So, WinGate running on a windows box can be configured (misconfigured) to permit bounced traffic from an external hostile source, anonymous communication with relay chat servers, or

anonymous web surfing. The "echo requests" scan in this detect were used to find live machines. Then the SYN scan to port 1080 was used to look for the WinGate proxy. This was followed by SYN packets to 3 other ports, presumably looking for trojans that might reside there.

There are false positives associated with the WinGate alerts. I discount them in this detect because there are also three attempts to communicate on three other ephemeral ports in this incident. Hostiles who look for trojaned ports are quite likely to be interested WinGate as well.

6) Correlations:
There are CERT incident notes for WinGate:
http://www.cert.org/incident_notes/IN-99-01.html
http://www.cert.org/vul_notes/VN-98.03.WinGate.html

White Hat has some information
http://www.whitehats.com/info/IDS175

There are CVEs from http://cve.mitre.org
CVE-1999-0290
The WinGate telnet proxy allows remote attackers to cause a denial of service via a large number of connections to localhost.
CVE-1999-0291
The WinGate proxy is installed without a password, which allows remote attackers to redirect connections without authentication.
CVE-1999-0441
Remote attackers can perform a denial of service in WinGate machines using a buffer overflow in the Winsock Redirector Service.
CVE-1999-0494
Denial of service in WinGate proxy through a buffer overflow in POP3.
CAN-1999-0657
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0657

There have been several GIAC practicums with detects and analysis of SubSeven.
http://www.sans.org/giactc/gcia.htm. c.f.

Terri Bidwell
Becky Bogle
Guy Bruneau
Tom Chmielarski
Robert Clark
Bradley Galvin
Kyle Nakamura

7) Evidence of Active Targeting:
This incident certainly evidenced active targeting.

The attack was "aimed" at 2 specific IP addresses.
And, attack was aimed only at Ports 8080, 1085, 7000, 8000, and 2008.

8) Severity:
Severity = (Criticality + Lethality) – ($CounterMeasures_{System}$ + $CounterMeasures_{Network}$)
Severity = (2 + 4) - (3 + 2)
Severity = -1

Comment: One of these IP addresses is to our user area disk farm. It is patched, and secure, and backed up every night. Nonetheless, it is a critical resource. I assign Lethality = 4.

9) Defensive Recommendation:
Damages: None of the targets were configured as proxy servers, and so no penetration occurred.
Check out the Disk Farm, to insure operating system integrity.

10) Multiple Choice Test Question:

Consider this tcpdump log:

```
06:13:54.831363 < 202.39.9.109.1094 > mynet.4.245.7000: S 2931551856:2931551856(0)
win 16384 <mss 1460,nop,nop,sackOK> (DF)

06:13:54.831648 < mynet.4.245.7000 > 202.39.9.109.1094: R 0:0(0) ack 2931551857 win 0
(DF)
```

Which statement is true?

a) This is UDP traffic.
b) This is crafted traffic because the number 2931551856 in the first packet, is incremented by 1 in the second packet to 2931551857.
c) The "0:0" element in the second packet confirms that this is crafted.
d) The target port, 7000, indicates that this is WinGate activity.
e) The address mynet.4.245 in this detect must be spoofed.
f) The 0:0 element and the "(DF)" element in the second packet are inconsistent with one another. This is more evidence of packet crafting.
g) None of the above.

Detect 8: Trolling for Trojans (and maybe ftp)

Summary:

From: 203.198.8.202 Several ephemeral ports starting with 4837

To: mynet.4.245 To the following ports

All these port numbers were targeted.

The trojan names in this list are by NO means exhaustive

21	ftp, but also Back Construction, Trojan AKA: Back Construction, Blade Runner, Cattivik FTP Server, CC Invader, Dark FTP, Doly Trojan, Fore, Invisible FTP, Juggernaut 42, Larva, MotIv FTP, Net Administrator, Ramen, Senna Spy FTP, server, The Flu, Traitor 21, WebEx, WinCrash Blade Runner, Doly Trojan, Fore, FTP trojan, Invisible FTP, Larva, WebEx, WinCrash
82	Unknown
83	Unknown, www.incidents.org identifies port 83 as the 31st most commonly targeted port over the last 30 days.
86	Unknown
110	POP3, but also ProMail trojan
666	Attack FTP, Back Construction, BLA trojan, Cain & Abel, NokNok, Satans Back Door - SBD, ServU, Shadow Phyre, th3r1pp3rz (= Therippers), Back Construction, doom Id Software, Cain & Abel, Satanz Backdoor, ServeU, Shadow Phyre
1010	surf, Doly Trojan
1128	Unknown
2000	Der Späher / Der Spaeher, Insane Network, Last 2000, Remote Explorer 2000, Senna Spy Trojan Generator, TransScout
2020	Unknown
2128	Unknown
3030	Unknown
4000	SkyDance
4080	Unknown
4128	Unknown
5050	Unknown
5080	Unknown
5128	Unknown
6000	The Thing
8000	Unknown, www.incidents.org identifies port 8000 as the 10th most commonly targeted port over the last 30 days.
8070	Unknown
8085	Unknown
8086	Unknown
8800	Unknown

8877 Unknown

```
[root@liar July05]# geektools 203.198.8.202
[whois.geektools.com]
Query:     203.198.8.202
Registry:  whois.apnic.net
Results:

inetnum:     203.198.0.0 - 203.198.31.255
netname:     NETVIGATOR
descr:       HongKong Telecom IMS
descr:       22/F, Tower II, Grand Central Plaza,
descr:       Shatin, Hong Kong.
country:     HK

[root@liar July05]# nslookup 203.198.8.202
202.8.198.203.in-addr.arpa      name = pcd092202.netvigator.com.
```

Observed Traffic:

#---- Portscan from: 203.198.8.202 ----------

```
Jul  5 22:14:43 203.198.8.202:4856 -> mynet.4.245:1128 SYN ******S*
Jul  5 22:14:46 203.198.8.202:4839 -> mynet.4.245:110 SYN ******S*
Jul  5 22:14:46 203.198.8.202:4840 -> mynet.4.245:82 SYN ******S*
Jul  5 22:14:46 203.198.8.202:4844 -> mynet.4.245:86 SYN ******S*
Jul  5 22:14:46 203.198.8.202:4862 -> mynet.4.245:3030 SYN ******S*
Jul  5 22:14:46 203.198.8.202:4865 -> mynet.4.245:4000 SYN ******S*
Jul  5 22:14:46 203.198.8.202:4867 -> mynet.4.245:4080 SYN ******S*
Jul  5 22:14:46 203.198.8.202:4868 -> mynet.4.245:4128 SYN ******S*
Jul  5 22:14:46 203.198.8.202:4870 -> mynet.4.245:5050 SYN ******S*
Jul  5 22:14:46 203.198.8.202:4872 -> mynet.4.245:5128 SYN ******S*
Jul  5 22:14:46 203.198.8.202:4873 -> mynet.4.245:6000 SYN ******S*
Jul  5 22:14:46 203.198.8.202:4887 -> mynet.4.245:8070 SYN ******S*
Jul  5 22:14:48 203.198.8.202:4841 -> mynet.4.245:83 SYN ******S*
Jul  5 22:14:49 203.198.8.202:4858 -> mynet.4.245:2020 SYN ******S*
Jul  5 22:14:52 203.198.8.202:4848 -> mynet.4.245:666 SYN ******S*
Jul  5 22:14:52 203.198.8.202:4853 -> mynet.4.245:1010 SYN ******S*
Jul  5 22:14:52 203.198.8.202:4857 -> mynet.4.245:2000 SYN ******S*
Jul  5 22:14:52 203.198.8.202:4871 -> mynet.4.245:5080 SYN ******S*
Jul  5 22:14:52 203.198.8.202:4881 -> mynet.4.245:8000 SYN ******S*
Jul  5 22:14:52 203.198.8.202:4891 -> mynet.4.245:8085 SYN ******S*
Jul  5 22:14:52 203.198.8.202:4892 -> mynet.4.245:8086 SYN ******S*
Jul  5 22:14:52 203.198.8.202:4897 -> mynet.4.245:8800 SYN ******S*
Jul  5 22:14:52 203.198.8.202:4900 -> mynet.4.245:8877 SYN ******S*
```

#---- snort logs from: 203.198.8.202 ----------

```
[**] ICMP Echo Request [**]
07/05-22:14:12.633205 203.198.8.202 -> mynet.4.184
ICMP TTL:102 TOS:0x0 ID:12547 IpLen:20 DgmLen:64
```

```
Type:8  Code:0  ID:60420   Seq:0  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP Echo Request [**]
07/05-22:14:14.613283 203.198.8.202 -> mynet.5.121
ICMP TTL:106 TOS:0x0 ID:16398 IpLen:20 DgmLen:64
Type:8  Code:0  ID:60420   Seq:0  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP Echo Request [**]
07/05-22:14:14.786049 203.198.8.202 -> mynet.5.140
ICMP TTL:106 TOS:0x0 ID:16748 IpLen:20 DgmLen:64
Type:8  Code:0  ID:60420   Seq:0  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP Echo Request [**]
07/05-22:14:14.830212 203.198.8.202 -> mynet.5.141
ICMP TTL:106 TOS:0x0 ID:16773 IpLen:20 DgmLen:64
Type:8  Code:0  ID:60420   Seq:0  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP Echo Request [**]
07/05-22:14:14.884810 203.198.8.202 -> mynet.5.144
ICMP TTL:106 TOS:0x0 ID:16828 IpLen:20 DgmLen:64
Type:8  Code:0  ID:60420   Seq:0  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

. . . . 21 qualitatively similar records . . . .

#---- tcpdump from src 203.198.8.202 --------------------------
 #   First some echo requests to a series of computers

```
22:14:12.633205 < pcd092202.netvigator.com > life.mynet.org: icmp: echo request (ttl 102, id 12547)
22:14:12.765982 < pcd092202.netvigator.com > life.mynet.org: icmp: echo request (ttl 102, id 12547)
```

 #   Then sphinx provides an echo reply

```
22:14:13.393261 < pcd092202.netvigator.com > sphinx.mynet.org: icmp: echo request (ttl 106, id 13818)
22:14:13.393589 < sphinx.mynet.org > pcd092202.netvigator.com: icmp: echo reply (DF) (ttl 255, id 31917)
```

 #   More echo requests to other computers

```
22:14:14.613283 < pcd092202.netvigator.com > luckyluke.mynet.org: icmp: echo request (ttl 106, id 16398)
22:14:14.746084 < pcd092202.netvigator.com > luckyluke.mynet.org: icmp: echo request (ttl 106, id 16398)
22:14:14.746974 < pcd092202.netvigator.com > luckyluke.mynet.org: icmp: echo request (ttl 106, id 16398)
22:14:14.786049 < pcd092202.netvigator.com > daffyduck.mynet.org: icmp: echo request (ttl 106, id 16748)
22:14:14.830212 < pcd092202.netvigator.com > mastermind.mynet.org:
```

. . . . 21 echo requests . . . .

 # sphinx echo replied, so he devotes some attention to it
 # This is a scan of ports, looking for Trojans.
 # The packets are mostly SYN-RST pairs
22:14:43.275489 < pcd092202.netvigator.com.4841 > sphinx.mynet.org.83: S 4051211602:4051211602(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 106, id 5287)
22:14:43.275778 < sphinx.mynet.org.83 > pcd092202.netvigator.com.4841: R 0:0(0) ack 4051211603 win 0 (DF) (ttl 106, id 31918)

22:14:43.677783 < pcd092202.netvigator.com.4856 > sphinx.mynet.org.1128: S 4051948009:4051948009(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 106, id 5498)
22:14:43.678054 < sphinx.mynet.org.1128 > pcd092202.netvigator.com.4856: R 0:0(0) ack 4051948010 win 0 (DF) (ttl 106, id 31919)

22:14:43.681830 < pcd092202.netvigator.com.4858 > sphinx.mynet.org.2020: S 4052039685:4052039685(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 106, id 5511)
22:14:43.682105 < sphinx.mynet.org.2020 > pcd092202.netvigator.com.4858: R 0:0(0) ack 4052039686 win 0 (DF) (ttl 106, id 31920)

22:14:43.689740 < pcd092202.netvigator.com.4860 > sphinx.mynet.org.2128: S 4052141457:4052141457(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 106, id 5528)
22:14:43.690015 < sphinx.mynet.org.2128 > pcd092202.netvigator.com.4860: R 0:0(0) ack 4052141458 win 0 (DF) (ttl 106, id 31921)

22:14:46.203148 < pcd092202.netvigator.com.4844 > sphinx.mynet.org.86: S 4051374189:4051374189(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 106, id 10752)
22:14:46.203415 < sphinx.mynet.org.86 > pcd092202.netvigator.com.4844: R 0:0(0) ack 4051374190 win 0 (DF) (ttl 106, id 31922)

22:14:46.204804 < pcd092202.netvigator.com.4839 > sphinx.mynet.org.pop3: S 4051132429:4051132429(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 106, id 10756)
22:14:46.205076 < sphinx.mynet.org.pop3 > pcd092202.netvigator.com.4839: R 0:0(0) ack 4051132430 win 0 (DF) (ttl 106, id 31923)

22:14:46.207188 < pcd092202.netvigator.com.4840 > sphinx.mynet.org.82: S 4051168581:4051168581(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 106, id 10758)
22:14:46.207463 < sphinx.mynet.org.82 > pcd092202.netvigator.com.4840: R 0:0(0) ack 4051168582 win 0 (DF) (ttl 106, id 31924)

 # sphinx listens to ftp, so there is a completed 3-way handshake
 # There are also several Trojans that use port 21
22:14:46.208835 < pcd092202.netvigator.com.4837 > sphinx.mynet.org.ftp: S 4051021474:4051021474(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 106, id 10759)
22:14:46.209119 < sphinx.mynet.org.ftp > pcd092202.netvigator.com.4837: S 863690456:863690456(0) ack 4051021475 win 9898 <nop,nop,sackOK,mss 1414> (DF) (ttl 255, id 31925)
22:14:46.408324 < pcd092202.netvigator.com.4872 > sphinx.mynet.org.5128: S 4052818077:4052818077(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 106, id 11169)

 # But it is tcpwrapped, so the connection was was reset
22:14:46.408606 < sphinx.mynet.org.5128 > pcd092202.netvigator.com.4872: R 0:0(0) ack 4052818078 win 0 (DF) (ttl 106, id 31926)

```
#     More SYN RST pairs
22:14:46.409947 < pcd092202.netvigator.com.4870 > sphinx.mynet.org.5050: S
4052711369:4052711369(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 106, id 11170)
22:14:46.410223 < sphinx.mynet.org.5050 > pcd092202.netvigator.com.4870: R 0:0(0) ack
4052711370 win 0 (DF) (ttl 106, id 31927)

22:14:46.411117 < pcd092202.netvigator.com.4865 > sphinx.mynet.org.polygenld: S
4052440792:4052440792(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 106, id 11171)
22:14:46.411401 < sphinx.mynet.org.polygenld > pcd092202.netvigator.com.4865: R 0:0(0) ack
4052440793 win 0 (DF) (ttl 106, id 31928)

22:14:46.412368 < pcd092202.netvigator.com.4867 > sphinx.mynet.org.4080: S
4052563589:4052563589(0) win 16384 <mss 1414,nop,nop,sackOK> (DF) (ttl 106, id 11172)
22:14:46.412658 < sphinx.mynet.org.4080 > pcd092202.netvigator.com.4867: R 0:0(0) ack
4052563590 win 0 (DF) (ttl 106, id 31929)

   . . . . 62 similar lines, several different ports. . . .

#     Syslog server: grep cd092202.netvigator.com /var/log/syslog
Jul  5 22:14:46 sphinx.mynet.org ftpd[29108]: refused connect from pcd092202.netvigator.com
```

1) Source of Trace:
My Laboratory

2) Detect was Generated by:
Site maintains 6 IDS sensors running tcpdump 24/7.

```
% tcpdump –w dumpFile_24hour
% snort –r dumpFile_24hour -c /etc/snort/snort.conf
```

At midnight, on a designated analysis workstation, process data using several shell, sed, awk, and perl scripts.
Gather all outside IP addresses and suspicious inside IP address of interest.
Obtain tcpdump data in separate files, for each "IP address of interest".

```
% tcpdump –r dump_24hour -vv "dst $outsider or src $outsider" \
        > tcpdump_IPaddress
% tcpdump -r dumpFile_24hour –vv -x \
        "dst $outsider or src $outsider" > tcpdump_hex_IPaddress
```

Snort Rule that generated this detect:
preprocessor portscan:

And, and info.rules:
```
alert icmp !$HOME_NET any -> any any (msg:"ICMP Echo Request"; itype: 8; icode: 0;)
```

3) Probability that Source Address was Spoofed:
Very Unlikely
Reason: Using "echo requests" the hostile found computers that were alive. He knew they were alive because of the "echo reply". Then he established (and tried to establish)

TCP 3 way handshake communication to various ports. All this activity requires that the hostile be actively listening.

4) Description of Attack:

From:	203.198.8.202 Several ephemeral ports starting with 4837
To:	mynet.4.245 To often trojaned ports

Average Interval:	2.3 seconds
Protocol:	ICMP and TCP
ID Numbers:	Unremarkable
Seq. Numbers:	Unremarkable
Ack. Numbers:	Unremarkable
TOS:	0x00
TTL:	102

This incident was a stimulus event (echo request) followed by a second stimulus (SYN).
This incident involved only "Normal TCP/IP" traffic.
This incident was fast (~ 26 hits/minute).
This incident involved a small volume of traffic. It scanned elements of an entire subnet, then targeted 1 workstation only.
This incident was a reconnaissance and exploitation of vulnerability as discovered
The target OS for this incident is a Solaris workstation.
The source OS for this incident is unknown.

5) Attack Mechanism:

Following an ICMP "Echo Request" scan, a SYN scan, perhaps by netcat or similar tool (good for scanning ports), was used to probe "Echo Reply" positive workstations. This detect was clearly an attempt to find trojaned computers because all the target ports were either ephemeral, or well associated with known trojan activity.

Many of these trojans work on windows 9x/NT/2000 workstations. So this was certainly a target. There were likely other operating system targets as well, as indicated by the shear number of probed ports.

The source operating system is unknown.

6) Correlations:

There are CERT advisories for trojaned programs:
http://www.cert.org/advisories/CA-1999-02.html
http://www.cert.org/JHthesis/Word6/Glossary.doc

And several CVEs that deal with trojans
CVE-2000-0663
The registry entry for the Windows Shell executable (Explorer.exe) in Windows NT and Windows 2000 uses a relative path name, which allows local users to execute arbitrary

commands by inserting a Trojan Horse named Explorer.exe into the %Systemdrive% directory, aka the "Relative Shell Path" vulnerability.
CVE-2000-0854
When a Microsoft Office 2000 document is launched, the directory of that document is first used to locate DLL's such as riched20.dll and msi.dll, which could allow an attacker to execute arbitrary commands by inserting a Trojan Horse DLL into the same directory as the document.
CVE-2000-1072
iCal 2.1 Patch 2 installs many files with world-writeable permissions, which allows local users to modify the iCal configuration and execute arbitrary commands by replacing the iplncal.sh program with a Trojan horse.
CVE-2000-1073
csstart program in iCal 2.1 Patch 2 searches for the cshttpd program in the current working directory, which allows local users to gain root privileges by creating a Trojan Horse cshttpd program in a directory and calling csstart from that directory.
CVE-2000-1074
csstart program in iCal 2.1 Patch 2 uses relative pathnames to install the libsocket and libnsl libraries, which could allow the icsuser account to gain root privileges by creating a Trojan Horse library in the current or parent directory.
CVE-2000-1163
ghostscript before 5.10-16 uses an empty LD_RUN_PATH environmental variable to find libraries in the current directory, which could allow local users to execute commands as other users by placing a Trojan horse library into a directory from which another user executes ghostscript.
CVE-2001-0289
Joe text editor 2.8 searches the current working directory (CWD) for the .joerc configuration file, which could allow local users to gain privileges of other users by placing a Trojan Horse .joerc file into a directory, then waiting for users to execute joe from that directory.

There is a CAN number for trojans. (CANs are Canditates for inclusion on the CVE list)
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660

Hacking Exposed, Scambray, McClure, and Kurtz, page 555-561, Second Edition, Osborne/McGraw Hill (2001)

7) Evidence of Active Targeting:
This incident certainly evidenced active targeting.
The attack was aimed a specific network. After finding, a living workstation, the attack was “aimed” at 1 specific IP addresses.
It probed for the presence of specific Trojans that might be listening to one of 25 ports.

8) Severity:
Severity = (Criticality + Lethality) – ($CounterMeasures_{System}$ + $CounterMeasures_{Network}$)
Severity = (2 + 4) - (5 + 2)

Severity = -1

Comment: This system has been patched, it has not been compromised by any trojans. The command lsof determined that sphinx does not listen to any unexplained or exotic ports.

9) Defensive Recommendation:
Damages: No penetration
Consider blocking some of these ports at the firewall.

10) Multiple Choice Test Question:

Consider the following trace:

```
[**] ICMP Echo Request [**]
07/05-22:14:12.633205 203.198.8.202 -> mynet.4.184
ICMP TTL:102 TOS:0x0 ID:12547 IpLen:20 DgmLen:64
Type:8  Code:0  ID:60420   Seq:0  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP Echo Request [**]
07/05-22:14:14.613283 203.198.8.202 -> mynet.5.121
ICMP TTL:106 TOS:0x0 ID:16398 IpLen:20 DgmLen:64
Type:8  Code:0  ID:60420   Seq:0  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ICMP Echo Request [**]
07/05-22:14:14.786049 203.198.8.202 -> mynet.5.140
ICMP TTL:106 TOS:0x0 ID:16748 IpLen:20 DgmLen:64
Type:8  Code:0  ID:60420   Seq:0  ECHO
```

Which statement is true.

a) These are tcpdump files alert logs
b) This is TCP traffic
c) The element “[**] ICMP Echo Request [**]” is part of the observed traffic.
d) The elements ”Type:8 Code:0” mean an Echo Request sent by host.

Bibliography

Network Intrusion Detection, An Analyst's Handbook, S. Northcutt, New Riders, (1999).

Intrusion Signatures and Analysis, S. Northcutt, M. Cooper, M Fearnow, and K. Frederick, New Riders, (2001).

Hacking Exposed: Network Security Secrets and Solutions, Second Edition, J. Scambray, S. McClure, G. Kurtz, Osborne/McGraw Hill, (2001).

TCP/IP Illustrated, Volume 1, WR Stevens, Addison-Wesley, (1994).

TCP/IP Illustrated, Volume 2, WR Stevens, Addison-Wesley, (1995).

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0657
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660
http://cve.mitre.org/cve
http://www.cert.org/advisories/CA-1999-02.html
http://www.cert.org/current/current_activity.html
http://www.cert.org/incident_notes/IN-2000-02.html
http://www.cert.org/incident_notes/IN-2001-07.html
http://www.cert.org/incident_notes/IN-99-01.html
http://www.cert.org/JHthesis/Word6/Glossary.doc
http://www.cert.org/vul_notes/VN-2000-03.html
http://www.cert.org/vul_notes/VN-98.03.WinGate.html
http://www.incidents.org
http://www.map2.ethz.ch/ftp-probleme.htm
http://www.nipc.gov/warnings/advisories/2000/00-056.htm
http://www.nipc.gov/warnings/advisories/2001/01-014.htm
http://www.sans.org/giactc/Becky_Bogle_GCIA.doc
http://www.sans.org/giactc/gcia.htm
http://www.sans.org/topten.htm
http://www.snort.org
http://www.whitehats.com/info/IDS175

Flag Burning

Introduction

This paper concerns the general problem of "Operating System Finger printing" and the role it plays in intrusive computer activity. I have been drawn to this topic by the astonishing observation in the "Analyze This" assignment of this practicum, that during the one-month period of analysis performed for Assignment 3, that there were 139 different exotic TCP flag combinations observed in just under 59,000 detects. In the GIAC "Intrusion Detection in Depth" course, one learns of SYN-FIN scans, NULL scans, Christmas Tree scans, and the role that those scans play in OS finger printing. Indeed, in the Assignment 3 data, these three mal-formed, but well-known, TCP flag combinations constituted some 45,000 of the exotic TCP flag settings observed. Those cases are indicated by bold in the following table. There remains however, some 14,000 detects with one of 137 other exotic TCP flag settings.

Alert Count	Mal Formed Flags	Alert Count	Mal Formed Flags	Alert Count	Mal Formed Flags	Alert Count	Mal Formed Flags
44511	****SF******	16	21S*R*AU	11	*1SF****	7	***FRPAU
7726	21S*****	15	**SF**AU	11	*1SF**AU	7	*1SF***U
4396	RESERVEDBITS	15	**SFRPA*	11	*1SFR*AU	7	*1SF**A*
460	********	15	21*FR***	11	2*SFR***	7	*1SFRPA*
94	***FR*A*	15	21S***AU	11	21*F**A*	7	*1SFRPAU
93	**S*R*A*	15	21S**P*U	11	21*FRP**	7	2*SF**A*
79	**SFRP*U	15	21S*R***	11	21S**PA*	7	21**RP**
37	***F****	15	21SF****	10	***FRP**	7	21*FR**U
28	****RP**	15	21SF*PA*	10	*1SFRP*U	7	21S***A*
28	**SFR**U	15	21SF*PAU	10	21**RPAU	7	21S**PAU
25	2*SFR*AU	15	21SFR**U	10	21*F***U	7	21S*RP**
25	21*F****	15	21SFR*AU	10	21*FR*A*	7	21S*RP*U
24	***FR***	**15**	**21SFRPAU**	10	21*FRPA*	7	21SF**AU
24	21*F*PA*	14	2*SFRPA*	10	21SF**A*	6	*****P*U
23	**SF***U	14	21S*R**U	9	*1SF*P**	6	***FRP*U
23	**SF*PAU	14	21S*RPA*	9	*1SF*PA*	6	**S*RP**
23	2*SFRP*U	14	21S*RPAU	9	*1SFR*A*	6	*1SFR***
23	21*FR*AU	13	*1SF*P*U	9	2*SFR*A*	6	21**R***
21	**S*RP*U	13	2*SF*PA*	9	2*SFRPAU	6	21S*R*A*
21	**SF*PA*	13	2*SF*PAU	9	21*F**AU	5	**S****U
21	2*SF***U	13	21*F*P**	9	21*F*P*U	5	**S**PA*
20	21*FRPAU	13	21SF***U	9	21*FRP*U	5	**S*RPA*
19	***F*P*U	13	21SFR*A*	9	21SF*P**	5	**S*RPAU
19	**SF**A*	12	*******U	9	21SFR***	4	**S**P**
19	**SF*P**	12	**S***AU	9	21SFRP**	4	**S**P*U

19	**SFR***	12	**SFR*AU	9	21SFRP*U	4	**S*R*AU
19	2*SF****	12	*1SFRP**	9	21SFRPA*	3	****R**U
18	****RP*U	12	2*SF**AU	8	**SFRP**	3	****R*AU
18	***FRPA*	12	2*SF*P*U	8	*1SFR**U	3	****RPAU
18	21**RPA*	12	21*F*PAU	8	2*SF*P**	3	***F***U
17	**SF*P*U	12	21S**P**	8	2*SFR**U	3	***FR**U
17	2*SFRP**	11	***F*P**	8	21**R**U	3	**S**PAU
17	21**R*AU	11	**S*R***	8	21**RP*U	2	***FR*AU
16	*1SF*PAU	11	**SFR*A*	8	21S****U	2	**S*R**U
16	21**R*A*	11	**SFRPAU	8	21SF*P*U		

This discussion will undertake to synopsize the general methods of intrusive network scanning, especially as they pertain to operating system finger printing. This will be followed by a synopsis of normal and abnormal TCP flagged traffic. Finally, I will briefly discuss the source code for tft.c, a tool that exercises a target workstation's TCP stack by passing it all possible combinations of 64 TCP flags.

Scanning

There exist tools for the "script kiddy" that permit the user to remain more or else clueless about what he or she is doing as they are merrily used against computer networks every day. And, while most of this intrusive activity is little more than noise in what passes for the life of serious computer security person; serious "hackers" (I prefer the term hostiles) and serious "hacking" involves competent committed intellectual effort devoted to breaking into, taking over, and using un-owned computer systems. It also involves the considerable effort and competence to write those no-brainer GUI, three clicks and attack tools used by the less skillful. The serious, and seriously intrusive hostile computer activity requires a plan, and such activity follows a reasonably well characterized pattern.

First there is target selection and open source information gathering, web browsing, whois, nslookup, social engineering, dumster diving, and the like. This is followed by a scanning phase, in which tools are used to map out the target network and resources, and an effort is made to determine the operating systems with their versions, obtain passwds, obtain dns zone transfers, learn the versions of installed software, enumerate shares, security policies, firewall rules, etc. From this, the hostile prepares maps of current IP addresses, and locations of networked services. Additionally, hostiles make an evaluation of the general level of system administration alertness (perhaps even cluefullness). Only then, is serious intrusive activity undertaken to compromise a computer, a series of computer, or an entire network. This is followed by one of many possible scenarios depending on the motivation of the intruder: perhaps intrusive tool installation, relay chat installation, web defacement, data theft, or DDOS slave configuration, back door creation, trojan installation, etc, etc. In this paper, I will focus on the scanning OS finger printing stage of this process.

The purpose of scanning is to gather information remotely about a network, computer, or service that will enable more intrusive computer activity by the intruder. Scanning starts out using tools that gather elementary information such as an enumerated list of subnet

address space and a list of live IP addresses. Scanning then proceeds to enumerate lists of available services and open ports that might be exploited. Enumerated knowledge of the open ports, by itself, may be enough to identify the operating system behind a live IP address. For example, netbios ports, 137, 138, and 139, can be indicative of Windows architecture and port 111 is the Solaris portmapper and indicates a Unix computer on an nfs network. Finally, scanning tools are used to determine operating systems types and versions so that specific vulnerabilities can be targeted in the invasive stages. Several tools are used for the scanning stage of a computer compromise.

Ping and ICMP Echo request Tools

The command ping sends an ICMP "echo request" packets to a targeted computer, which if alive will return an "echo reply". Things get more complicated if there is an intermediate firewall which blocks ICMP echo requests. The source computer might see a "destination unreachable" or no message what so ever. Careful interpretation of this traffic permits the hostile to map out IP address in the target network. This task can be simplified by the use of automated ping tools that ping networks through broadcasts to blocks of 255 IP addresses (or larger) at once, e.g. fping and gping (http://packetstormsecurity.org/Exploit_Code_Archive/fping.tar.gz) (http://www.hackingexposed.com/tools/tools.html). These tools, used with various combinations of switches, permit the hostile to search a range of network IP addresses for accessible (pingable) computers. The tool nmap –sP (http://www.insecure.org/nmap) can also be used to ping sweep a network by using a starting IP address and netmask. These are Unix tools. But there are Windows tools as well, pringer (http://www.nmrc.org/files/snt), WS_ping (http://www.ipswitch.com), and netscan (http://www.nwpsw.com). The tool icmpenum (http://www.nmrc.org/files/sunix/icmpenum-1.1.1.tgz) permits sending not only echo requests, but time stamp requests, and ICMP info requests as well. This tool used in these modified ways, may enable intruders to get around firewalls that block direct ICMP echo requests.

NMAP

As discussed above, the tool nmap, can be used to perform ping sweeps, but it is much more versatile than that. Nmap can be used for port scanning (nmap -sP), TCP ping or ACK scanning directed at a specific port (e.g. to scan smtp, nmap -pT25). It provides port numbers, service names, and service owners. Additionally it can be used for TCP stealth scans (nmap -sS), UDP port scans (nmap -sU), stealth FIN scans (nmap -sF), and RPC/identd scans (nmap -sR). It has an operating system determination mode using TCP fingerprinting techniques (nmap -O), a decoy mode to hide scans by simultaneous bombardment of the target or IDS with bogus packets (nmap -D), a paranoid mode to slowly scan networks using large time intervals between packets (nmap -T), you can choose whether or not to do DNS resolution (nmap -n), you can send results to log files in one of several formats (nmap -iL), and you can specify the network interface or source address (nmap -S). You can use namp to fragment TCP packets in order to attempt evasion of the firewall (nmap -f). You can use nmap to hide the true source IP identity of a portscan by going through a misconfigured proxy ftp server (nmap -b) in a so-called ftp-

bounce attack. This makes a portscan virtually untraceable. Of course you can scan IP addresses or entire networks. It is the Rolls Royce of hacker tools.

Hping
The tool hping (http://www.kyuzz.org/antirez/hping.html) is a TCP ping/port scanning utility, which permits the user to craft elements of the TCP packet. In addition it supports UDP, ICMP, and RAW-IP protocols. According to its home page it is useful for testing firewalls, port scanning, network testing, TOS MTU and fragmentation testing, remote OS finger Printing, and TCP/IP stack auditing.

NETCAT
Netcat (or nc) is a basic all-purpose TCP/UDP port scanning tool. It does arbitrary portscanning, provides verbose output, provides timeouts, and it simultaneously targets ranges of IP addresses to enumerate lists of open ports.

Strobe
Strobe is an old tool (ftp://ftp.freeBSD.org/pub/FreeBSD/ports/packages/security/strobe-1.06.tgz). It is a TCP scanner. It can be used to obtain banners, and generally is used to enumerate a list of open ports on target IP addresses.

UDP_Scan
Complementary to strope is udp_scan (http://wwdsilx.wwdsi.com). It has a reputation as one of the best udp port scanning tools available. It is part of the satan/saint security vulnerability tool, and its use, fortunately, generates a satan alert in most IDS rule sets.

Windows based scanners
Of course, there are windows equivalents to these tools as well. NetScanTools Pro 2000 (http://www.netscantools.com/nstdownload.html) permits DNS queries, nslookup, dig, whois, ping sweeps, NetBIOS scans, SNMP scans, etc. You can perform simultaneous multiple component scans on the same network. SuperScan (http://www.foundstone.com/rdlabs/termsofuse.php?filename=superscan.exe) is another TCP scanner (freeware). There are several others.

TCP Communications and TCP Flags

TCP is an internet communication transport layer protocol that provides a reliable, bi-directional, error correcting service between two computers. Through the assignment of identifying session ID numbers, sequence numbers, and acknowledgement numbers to individual fragmented packets, TCP while not guaranteeing error free transmission of information does guarantee that all the pieces of a conversation will be transmitted, accepted, reassembled, and confirmed as they are transmitted from the source to the destination computer. The information in this section is synopsized from Stevens (1994, 1995) and Northcutt et al (2001).

To achieve this result, the TCP header, immediately following the IP header contains both the transmitted data as well as the auditing and descriptive information. In particular, the

TCP header contains the source and destination ports, the sequence and acknowledgement numbers, 32 bits of data offset, windows size, and special "flags" that categorize the type of TCP packet being sent. There are checksums, urgent pointers (if necessary), options, and finally the actually transmitted data.

In order to organize this traffic, TCP datagrams are identified with "flags" that identify the datagram as containing any one of several kinds of information:

Flag	Flag Description
SYN	Synchronize Sequence Numbers, initiates a connection
ACK	Set the Acknowledgement Number (Acknowledges the receipt of information)
PSH	Push data from the source to the target
URG	Alert target that the urgent pointer is set
FIN	Terminate connection, no more data
RST	Reset the connection

In addition, flags can be sent in certain (but not all) combinations to efficiently convey multiple bits of information from the source to the destination.

Flag or Flag Combination	Description
SYN-ACK	Synchronize the Sequence number and sets the Acknowledgement number
RST-ACK	Sets the Acknowledgement number and Resets the connection
FIN-ACK	Sets the Acknowledgement number and ends the session gracefully
FIN-PSH-ACK	Pushes additional data, Sets the Acknowledgement number, and ends the connection gracefully
URG-ACK	Sets the Acknowledgement number and set the urgent pointer
PSH-ACK	Sets the Acknowledgement number and push additional data

Flags and flag combinations other than these are at least quite uncommon, and in most cases improper, in that they do not conform to the RFC 793, which specifies the requirements for TCP communications.

The following figure, illustrates the proper structure of a TCP "SYN packet" with some data. The raw hex TCP header is indicated on the top row of the spreadsheet fragment shown below. The particular data for this header is parsed in the subsequent rows of the spreadsheet. Data in rows above the "blue row" indicate bit, hex, and byte counts of the TCP header. Cells in rows below the "blue row" contain the data associated with the TCP datagram shown on the first row, and are shown in binary, hex, and decimal. (I apologize for difficulty in reading this figure, it was built on a 21 inch monitor, and is difficult to read when pasted into a word processor. When printed, this page is much easier to read.)

TCP Header

03fe0016823c156b0000000060022238cf360000020405b45555

Bit Count	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Hex Count	0				1				2				3				4				5				6				7			
Byte Count	tcp[0]								tcp[1]								tcp[2]								tcp[3]							
Description	16 bit Source Port Number																16 bit Destination Port Number															
Binary	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0
Hex	0				3				f				e				0				0				1				6			
Decimal	1022																22															

Bit Count	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Hex Count	8				9				10				11				12				13				14				15			
Byte Count	tcp[4]								tcp[5]								tcp[6]								tcp[7]							
Description	32 bit Sequence Number																															
Binary	1	0	0	0	0	0	1	0	0	0	1	1	1	1	0	0	0	0	0	1	0	1	0	1	0	1	1	0	1	0	1	1
Hex	8				2				3				c				1				5				6				b			
Decimal	2184975723																															

Bit Count	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
Hex Count	16				17				18				19				20				21				22				23			
Byte Count	tcp[8]								tcp[9]								tcp[10]								tcp[11]							
Description	32 bit Acknowledgement Number																															
Binary	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hex	0				0				0				0				0				0				0				0			
Decimal	0																															

Bit Count	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
Hex Count	24				25				26				27				28				29				30				31			
Byte Count	tcp[12]								tcp[13]								tcp[14]								tcp[15]							
Description	4 bit Data Offset				6 bit Reserved							URG	ACK	PSH	RST	SYN	FIN	16 bit Window Size															
Binary	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	1	0	0	0
Hex	6				0				0				2				2				2				3				8			
Decimal	6				0							0	0	0	1	0		8760														

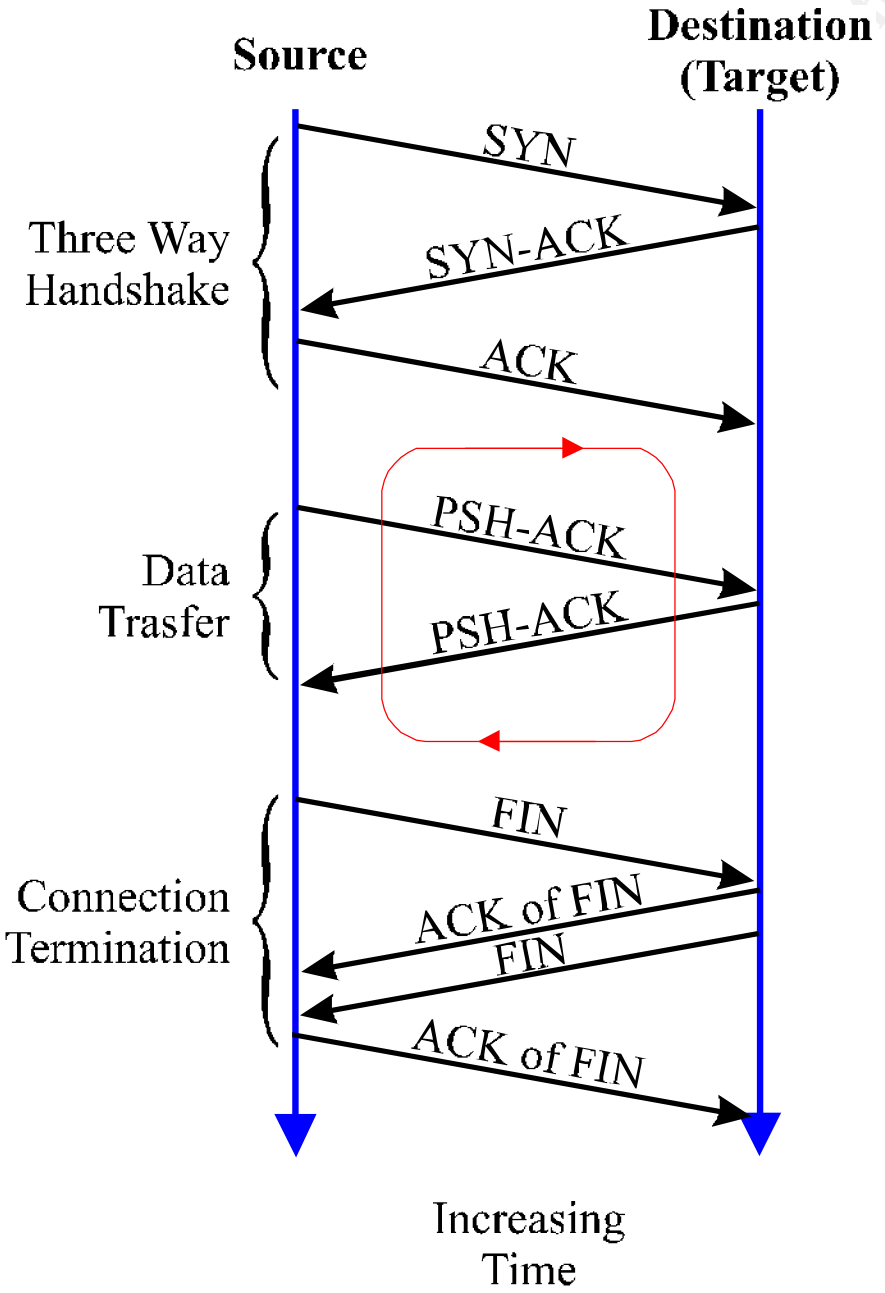
Bit Count	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
Hex Count	32				33				34				35				36				37				38				39			
Byte Count	tcp[16]								tcp[17]								tcp[18]								tcp[19]							
Description	16 bit Check Sum																16 bit Urgent Pointer															
Binary	1	1	0	0	1	1	1	1	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hex	c				f				3				6				0				0				0				0			
Decimal	53046																0															

Bit Count	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
Hex Count	40				41				42				43				44				45				46				47			
Byte Count	tcp[20]								tcp[21]								tcp[22]								tcp[23]							
Description	Options (If Any)																															
Binary	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	0	0
Hex	0				2				0				4				0				5				b				4			
Decimal	0				2				0				4				0				5				11				4			

Bit Count	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
Hex Count	48				49				50				51				52				53				54				55			
Byte Count	tcp[24]								tcp[25]								tcp[26]								tcp[27]							
Description	Data (If Any)																															
Binary	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1																
Hex	5				5				5				5																			
Decimal	5				5				5				5																			

TCP communication is divided into 3 stages. First a "3-way handshake" is established between a source and a destination computer. In this stage, the source initiates communication with a SYN packet (which initiates and synchronizes a sequence

number), the destination returns a SYN-ACK (that synchronizes its own sequence number, and sets an acknowledgement number), finally the source returns an ACK packet (that resets the acknowledgement number). All this occurs before the transfer of any data. In the second stage, the two computers send PSH, PSH-ACK packets, and if necessary RST-ACK packets back and forth as data is transferred between computers. Finally, the communication is taken down gracefully by a series of FIN, and FIN-ACK packets, or ungracefully by a RST packet. This is illustrated in the next figure. Many such diagrams, clearly illustrating variations on TCP communication, can be found in the fine series of books by Stevens (1994, 1995).



Schematic of a normal TCP communication session

RFC 793 specifies that there are 6 possible flags that can be set in a TCP datagram at tcp[13]. As indicated above, they can be sent in a "few" different combinations to accelerate communication of multiple bits of information in the same packet. Because 6 bits are required to specify all the flags in a datagram, there are $2^6 = 64$ possible combinations of flags that are possible to send from a source to a target. Add to this the 2 most significant bits in tcp[13], that is two of the so-called "reserved bits" which have been set aside for future use there are $2^8 = 256$ possible settings that can, in principle, be sent from a source to a target. The 4 least significant bits in tcp[12], also reserved bits, provide additional possibilities to created mal-formed packets. Most of these combinations are not allowed under the rubric of RFC 793. Indeed, most of these flag settings are logically inconsistent. For example, if the flags SYN-FIN are set in the same packet, this directs the target to both initiate the first step in a TCP 3-way connection and terminate that connection before it is established. Normal TCP traffic does not employ these improper flag settings. So, of course, hostile computer activity employs such traffic all the time.

The RFCs were designed to enumerate the rules for normal traffic. Only limited attention is devoted to all possible ways in which that traffic can be mal-formed. In general, the rule for a target computer when presented with a mal-formed TCP flag settings, is to return a RST. And usually, this is what happens. Unfortunately, some implementations of the TCP/IP implement the RFC standards in ways that are not uniform across platforms. For example, some vendors employ the "reserved bits" for ECN, but, some vendors simply do not get all the details right in their implementations of TCP/IP. The use of reserved bits for specialized application like ECN is a perfectly appropriate; on the other hand, exercising the reserved bits to scan targets for OS finger printing information is hostile activity.

Destinations controlled by certain operating systems respond to packets with improper flag settings differently from those controlled by other operating systems. Some operating systems are unable to process mal-formed packets properly, resulting if a frozen TCP communications session. In the worst cases, special combinations of flags at certain stages of a TCP communication session kill processes and daemons on the target computer or will even freeze the computer. For example, in the WinNuke vulnerability, unpatched Windows systems will "blue Screen" when a TCP packet is sent to an open port, the URG flag is set, and the urgent values is set to 3. Several operating systems can be finger printed by closely monitoring the returned flags when specially crafted packets with mal-formed flag combinations are sent to targets. See examples in the next section.

TCP Scans and OS Finger Printing Techniques

The simplest way of OS finger printing is by telneting to the target IP address and reading the banner (no login or passwd required). By telneting to the appropriate target ports, this technique can also be used to determine the version numbers of some services (ftp, smtp). Responsible system administrators, having caught on to this, have long ago starting removing, disabling, or decoying banners. In order to perform OS finger printing against a securely maintained network, a hostile must resort to scanning.

TCP scanning techniques are widely used to gather information about networks by hostiles. In particular, several TCP scanning techniques are useful for "Operating System Finger Printing". Most of this information in this section derives from the GIAC Intrusion Detection Course books, Scambray et al. (2001), and http://www.insecure.org/nmap/nmap-fingerprinting-article.html. The last paper, written by "Fyodor" the author of nmap contains the first serious reference to OS finger printing using scanning techniques.

TCP Connect Scan
In this scan, a full 3-way TCP connection is established and immediately closed. To be successful it must repeat this process on all ports for which information is desired.

TCP SYN Scan
In this more stealthy scanning technique scan, the source sends SYN packets to ports on the target. No other packets are sent. If the target is listening on the target port, the target returns a SYN-ACK packet, otherwise the target returns a RST. This information permits the source to enumerate IP addresses and open ports on the target. This scan is sometimes called a "half-open scan" or simply a "half-scan".

TCP FIN Scan
Isolated FIN packets targeting Unix boxes will result in a RST returned packet on closed ports. Open ports do not respond according to RFC 793. Unfortunately, windows NT responds with a FIN-ACK other operating systems return a RST from open ports. Thus this technique can be employed to perform OS finger printing.

TCP Xmas Tree Scan
When a source sends a "Christmas Tree" scan it sends an out of spec FIN-URG-PSH packet. The target will return a RST if the port is closed.

TCP Null Scan
A null scan is a TCP packet with no flags set. Again, because this is an out of spec packet, closed ports should return a RST packet. Some operating systems do not.

TCP ACK Scan
"ACK only" scans are used to probe firewalls. ACK packets will pass a "stateless" firewall, as the firewall is configured to permit "established TCP traffic", and so it interprets an ACK packet as part of an established connection. On the other hand, a "stateful" firewall, one that monitors entire communications sessions, discards

information regarding that communication only at the end of a complete session. So, a stateful firewall will block an isolated ACK packet at the firewall.

TCP Window Size Scan
Some Unix operating systems report TCP windows sizes in varying ways. This can be used to OS fingerprint AIX and FreeBSD.

TCP RPC Scan
On Unix systems, RPC ports are revealed by communication to various RPC ports the port numbers of which are OS specific. This technique is used for OS fingerprinting and the enumeration of available RPC ports.

Bogus Flag Probes
When a TCP SYN packet is sent with an undefined TCP "flag", the appropriate response for the target is not to respond. Unfortunately, Linux will respond with the bogus flag set. Other operating systems reply with RST.

Initial Sequence Number Sampling
Different operating systems employ different patterns in the ISN values upon the initiation of a 3-way TCP connection. This pattern can be deduced by multiple TCP queries. Old Unix boxes use 64K, Newer Solaris, Irix, freeBSD and others boxes use random increments, Windows uses an ISN incremented by a small time interval, and some boxes use a constant ISN (some 3com hubs, Apple LaserWriter printers). Even the random number generators can be characterized by their standard deviation and means and other arithmetic tests.

ACK Value
The acknowledgement numbers returned by closed ports, are normally are the same as the ISN for FIN, PSH, or URG flags. Windows uses ISN + 1. Additionally, Windows responds with an inconsistent ACK number when an open port is presented with a SYN, FIN, URG or PSH.

Don't Fragment Bit
Whether or not a TCP packet sets the DF flag by default is OS specific. And some OSs enable the DF bit only under some unique circumstances. Monitoring the use of this bit, reveals information about the OS version.

ICMP Error Message Quenching
According the RFC 1812, ICMP error messages are returned at a limited rate. That rate varies among OSs. For example, Linux sends only 80 destination unreachable messages in 4 seconds before imposing 0.25 second penalty. By sending a large number of UDP packets to a high number port, and counting the number of unreachable messages returned per time interval can be used for OS finger printing.

ICMP Message Quoting

ICMP error messages return a quoted portion of the ICMP message that caused the error. The length of the quoted returned segment is OS dependent. Usually, the error message returns 8 bytes, Solaris returns more and Linux returns even more.

ICMP Error Message Echoing Integrity
Some stacks alter the IP headers when returning ICMP error messages. These alterations are OS specific.

Type of Service
In ICMP port unreachable messages, the TOS is usually 0x00. Linux uses 0xc0.

Fragmentation handling
Different operating systems process overlapping fragments differently as they reassemble the fragmented packets. Some over write data, others give precedence to older data.

TCP Options
There is wide variation in how different operating systems handle TCP options. As options, not all implementations even handle all of them, they are placed in the options portions of the TCP header in different order, and different operating systems use different vales for different options. If a TCP packet is sent with all options set, the target will respond with only the supported options set in the return packet.

There are additional techniques that may be useful for OS finger printing. However, they can be counter-productive for the hostile. For example:

Exploit Chronology
All the techniques listed above are unable to distinguish among the various Windows flavors. The reason for this is that the Windows stack has remained unaltered as the technology was passed unchanged from Windows 95 -> Windows 98 -> Windows NT. Of course "Fyodor" is contemptuous of this state of affairs; because to him, it seems to indicate corporate engineering sloth. ("Marketing" in the Redmond office probably considers it a "feature".) In any event, "Fyodor" does not despair for long. His solution is to bombard the Windows box with a sequence of well-known and older windows attack tools. (Ping-o-death, WinNuke, etc.) Following each attack, he suggests sending additional packets to the target to determine its alive/dead status. Of course, the attacker must have a large collection of attack tools and a reliable database informing him which version of windows is vulnerable to which attack. This technique has the advantage of not only learning the Windows version, but also the Service pack release number for the target as a bonus. I am not at all sure whether or not this method is, at root, more motivated by some kind of political agenda, and it undeniably would be a fine DOS plan, but to me it seems relatively un-useful as a "stealthy OS finger print" tool.

SYN Flood Resistance
After establishing a certain number of partially open connections derived from SYN packets arriving from forged IP addresses, some operating systems will stop accepting

new connections. Most operating systems will accept only 8 partially open connections. So, by establishing 8 such connections, and probing the target for evidence of its willingness to accept more, the source use this information to characterize the OS. While more subtle that the "Exploit Chronology" this technique also suffers from being a bit to high profile to be useful. Neither of these techniques are used in the nmap code built by "Fyodor".

Finally there are other composite "wonder tools" that combine many of these techniques into one stand-alone tool. Cheops is a network mapping tool, with ping, traceroute, portscanning, a queso operating system detection tool, all rolled up into a single package with GUI. (http://www.marko.net/cheops) And, there are others in a similar vein.

tft.c

Queso

Queso (CERT® Incident Note IN-98.04, http://www.cert.org/incident_notes/IN-98.04.html, GIAC, Intrusion Detection 1-3, pp178-179) is not a technique, so much as a stand-alone intrusion package. It is an old tool, and these days most operating systems and IDS rule sets can prevent its successful use. The purpose of queso is to determine the operating system finger print of a target computer. And, it achieves this result by sending TCP packets with mal-formed flag settings and closely monitoring the returned packets.

NMAP

As discussed earlier, the nmap tool is a remarkably versatile and useful tool for hostile computer activity. (I suppose it even has some legitimate uses.) In the context of this section, one of its most useful features is that it provides the best OS finger printing tool available to the hostile community. This is a direct quote from "fyodor", http://www.insecure.org/nmap/nmap-fingerprinting-article.html .

> "We use the command:
> nmap -sS -F -o transmeta.log -v -O www.transmeta.com//24
>
> This says SYN scan for known ports (from /etc/services), log the results to 'transmeta.log', be verbose about it, do an OS scan, and scan the class 'C' where www.transmeta.com resides. Here is the gist of the results:
>
> neon-best.transmeta.com (206.184.214.10) => Linux 2.0.33-34
> www.transmeta.com (206.184.214.11) => Linux 2.0.30
> neosilicon.transmeta.com (206.184.214.14) => Linux 2.0.33-34
> ssl.transmeta.com (206.184.214.15) => Linux unknown version
> linux.kernel.org (206.184.214.34) => Linux 2.0.35
> www.linuxbase.org (206.184.214.35) => Linux 2.0.35
> (possibly the same machine as above)"

As an exercise for the student, I ran nmap against one computer only, on mynet.org with

the OS finger print options set:

```
[root@liar ids]# /usr/bin/nmap -sS -F -v -O mynet.org.4.235

Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Host sunder.mynet.org (mynet.org.4.235) appears to be up ... good.
Initiating SYN half-open stealth scan against sunder.mynet.org (mynet.org.4.235)
Adding TCP port 111 (state open).
Adding TCP port 32777 (state open).
Adding TCP port 4045 (state open).
Adding TCP port 2049 (state open).
Adding TCP port 32773 (state open).
Adding TCP port 110 (state open).
Adding TCP port 6112 (state open).
Adding TCP port 515 (state open).
Adding TCP port 514 (state open).
Adding TCP port 6000 (state open).
Adding TCP port 22 (state open).
Adding TCP port 25 (state open).
Adding TCP port 143 (state open).
Adding TCP port 7100 (state open).
Adding TCP port 32772 (state open).
Adding TCP port 32771 (state open).
Adding TCP port 748 (state open).
Adding TCP port 2766 (state open).
The SYN scan took 0 seconds to scan 1062 ports.
For OSScan assuming that port 22 is open and port 1 is closed and neither are firewalled
Interesting ports on sunder.mynet.org (mynet.org.4.235):
(The 1044 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
25/tcp     open        smtp
110/tcp    open        pop-3
111/tcp    open        sunrpc
143/tcp    open        imap2
514/tcp    open        shell
515/tcp    open        printer
748/tcp    open        ris-cm
2049/tcp   open        nfs
2766/tcp   open        listen
4045/tcp   open        lockd
6000/tcp   open        X11
6112/tcp   open        dtspc
7100/tcp   open        font-service
32771/tcp  open        sometimes-rpc5
32772/tcp  open        sometimes-rpc7
32773/tcp  open        sometimes-rpc9
32777/tcp  open        sometimes-rpc17

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=47758 (Worthy challenge)

Sequence numbers: 9B5770BB 9B57AA29 9B598137 9B59C29E 9B5A13EF 9B5BBB77
Remote OS guesses: Solaris 2.6 - 2.7, Solaris 7
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 4 seconds
```

That was impressive, and more to the point, the results are right. At the target end, on sunder.mynet.org, here is a very abbreviated response as determined by tcpdump. On sunder.mynet.org, this nmap triggered tcpdump generated nearly 1685 packets.

```
/opt/sbin/tcpdump -vv "dst mynet.org.4.142 or src mynet.org.4.142"

14:03:44.489430 < arp who-has liar.mynet.org tell gwcpc.mynet.org
14:03:44.606863 < liar.mynet.org > sunder.mynet.org: icmp: echo request
14:03:44.606903 < sunder.mynet.org > liar.mynet.org: icmp: echo reply (DF)
14:03:44.607030 < liar.mynet.org.56399 > sunder.mynet.org.http: . 765984771:765984771(0)
ack 2838295973 win 1024
14:03:44.607067 < sunder.mynet.org.http > liar.mynet.org.56399: R
2838295973:2838295973(0) win 0 (DF)
14:03:44.908789 < liar.mynet.org.ircs > sunder.mynet.org.731: udp 88 (DF)
14:03:44.909003 < sunder.mynet.org.731 > liar.mynet.org.ircs: udp 156 (DF)
14:03:44.912298 < liar.mynet.org.56379 > sunder.mynet.org.135: S 500425597:500425597(0)
win 1024
14:03:44.912380 < liar.mynet.org.56379 > sunder.mynet.org.4132: S 500425597:500425597(0)
win 1024
14:03:44.912458 < liar.mynet.org.56379 > sunder.mynet.org.1374: S 500425597:500425597(0)
win 1024
14:03:44.912538 < liar.mynet.org.56379 > sunder.mynet.org.6668: S 500425597:500425597(0)
win 1024
14:03:44.912616 < liar.mynet.org.56379 > sunder.mynet.org.6142: S 500425597:500425597(0)
win 1024
14:03:44.912697 < liar.mynet.org.56379 > sunder.mynet.org.10005: S
500425597:500425597(0) win 1024
14:03:44.912775 < liar.mynet.org.56379 > sunder.mynet.org.1411: S 500425597:500425597(0)
win 1024
14:03:44.912853 < liar.mynet.org.56379 > sunder.mynet.org.559: S 500425597:500425597(0)
win 1024
14:03:44.912932 < liar.mynet.org.56379 > sunder.mynet.org.1664: S 500425597:500425597(0)
win 1024
14:03:44.913012 < liar.mynet.org.56379 > sunder.mynet.org.197: S 500425597:500425597(0)
win 1024
14:03:44.913060 < sunder.mynet.org.135 > liar.mynet.org.56379: R 0:0(0) ack 500425598 win
0 (DF)

    . . . . 1685 total number of lines . . .

14:03:48.451736 < liar.mynet.org.56381 > sunder.mynet.org.ssh: R
2683758064:2683758064(0) win 0 (DF)
14:03:48.481523 < liar.mynet.org.56382 > sunder.mynet.org.ssh: S
2683758064:2683758064(0) win 1024
14:03:48.481564 < sunder.mynet.org.ssh > liar.mynet.org.56382: S
3590933174:3590933174(0) ack 2683758065 win 9112 <mss 53
6> (DF)
14:03:48.481733 < liar.mynet.org.56382 > sunder.mynet.org.ssh: R
2683758065:2683758065(0) win 0 (DF)
14:03:48.511529 < liar.mynet.org.56383 > sunder.mynet.org.ssh: S
```

```
2683758065:2683758065(0) win 1024
14:03:48.511569 < sunder.mynet.org.ssh > liar.mynet.org.56383: S
3590940072:3590940072(0) ack 2683758066 win 9112 <mss 53
6> (DF)
14:03:48.511737 < liar.mynet.org.56383 > sunder.mynet.org.ssh: R
2683758066:2683758066(0) win 0 (DF)
14:03:48.541529 < liar.mynet.org.56384 > sunder.mynet.org.ssh: S
2683758066:2683758066(0) win 1024
14:03:48.541570 < sunder.mynet.org.ssh > liar.mynet.org.56384: S
3590965818:3590965818(0) ack 2683758067 win 9112 <mss 53
6> (DF)
14:03:48.541741 < liar.mynet.org.56384 > sunder.mynet.org.ssh: R
2683758067:2683758067(0) win 0 (DF)
14:03:48.571537 < liar.mynet.org.56385 > sunder.mynet.org.ssh: S
2683758067:2683758067(0) win 1024
14:03:48.571579 < sunder.mynet.org.ssh > liar.mynet.org.56385: S
3591009122:3591009122(0) ack 2683758068 win 9112 <mss 53
6> (DF)
14:03:48.571749 < liar.mynet.org.56385 > sunder.mynet.org.ssh: R
2683758068:2683758068(0) win 0 (DF)
14:03:49.601451 < arp who-has sunder.mynet.org tell liar.mynet.org
14:03:49.601492 < arp reply sunder.mynet.org is-at 8:0:20:b0:32:e4
```

Only these flag settings were culled from the snort alerts that derived from this detect.

```
 12  ******S*          Normal flags
  4  **U*P*SF          Mal-formed flags
  4  ***A****          Normal flags
  2  **U*P**F          Mal-formed flags
      2    ***A**S*            Normal flags
```

All these flags were culled from all the TCP packets exchanged between sunder.mynet.org and liar.mynet.org

```
855  ******S*          Normal flags
765  ***A*R**          Normal flags
 25  *****R**          Normal flags
 22  ***A**S*          Normal flags
  3  ***A****          Normal flags
  2  **U*P*SF          Mal-formed flags
  2  ********          Mal-formed flags
  1  **U*P**F          Mal-formed flags
  1  *2****S*          Mal-formed flags
```

Tools such as queso and nmap do not explain the very large number of exotic flags seen in the thirty day "Analyze This" portion of this practicum alluded to I the opening paragraph of this paper. Both queso and nmap do indeed send exotic flag combinations to prospective victim-targets, however they do not send the very large range of mal-formed flag settings that are observed in the OOS alert files. Something else is at work here.

Following some web searching, I found an interesting tool at http://rootshell.com/archive-

j457nxiqi3gq59dv/199807/tft.c.html. It can also be found at http://packetstormsecurity.org/new-exploits/tft.c. The tool, tft.c, was apparently built by Lamont Granquist, and posted onto rootshell.com in 1998. It is advertised as a

> "TCP Flag Test – 'excersizes' a machines TCP/IP stack by passing it all combinations of 64 TCP flags and seeing which flags are usable to determine which ports on the machine are open or not"

Of course, I do not know that the program tft.c was used to generate the oos detects recorded at http://www.research.umbc.edu/~andy or not. It seems clear to me, that if not this tool, then a tool very much like this tool was used. It is quite clear that if this tool were used against several dozen known operating systems, the returned packets would yield a wealth of information that a motivated hostile could use for operating system finger printing. This information could be cataloged into a database that would provide a superior tool for network scanning.

I made a moderately serious effort to compile and test this program. The comments in the code specifically suggest that the code compiles on NetBSD, and perhaps NetBSD only. I do not have immediate access to any freeBSD flavors. It failed to compile it on Solaris, Irix and Linux. In any event, I am reluctant to exercise this kind of software on my net out of pure excessive caution; so I did not try all that hard.

Here is an abbreviated outline of what I think is going on in the code. The crucial fragments of the code are presented here as something like pseudo-code in order to clarify the function of the program.

Of course, there is a main:

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

It initializes ports

```
  initialize();
```

It gathers two port numbers from the command line

```
  oport = atoi(argv[2]);
  cport = atoi(argv[3]);
```

It resolves an IP address, also obtained from the command line

```
  if(resolve_host(argv[1], &dst_inaddr) < 0)
```

Something to do with sockets, apparently this comes from libpcap

```
  sethdrinclude(sd);
```

It builds an IP header

```
char *ip_field[] = { "ip_vhl"  , "ip_tos"  , "ip_len"  , "ip_len"  ,
                     "ip_id"   , "ip_id"   , "ip_off"  , "ip_off"  ,
                     "ip_ttl"  , "ip_p"    , "ip_sum"  , "ip_sum"  ,
                     "ip_src"  , "ip_src"  , "ip_src"  , "ip_src"  ,
                     "ip_dst"  , "ip_dst"  , "ip_dst"  , "ip_dst"  ,
                     "th_sport", "th_sport", "th_dport", "th_dport",
                     "th_seq"  , "th_seq"  , "th_seq"  , "th_seq"  ,
                     "th_ack"  , "th_ack"  , "th_ack"  , "th_ack"  ,
                     "th_xoff" , "th_flags", "th_win"  , "th_win"  ,
```

```
                "th_sum"  , "th_sum"  , "th_urp"  , "th_urp" };
```
Then opens a packet sniffer (kludged from tcpdump)
```
open_pkt_sniffer(cmdbuf, 60, NULL)
```
It exercises a for loop, once for each of 64 possible flag combination
```
for(i = 0; i<64; i++) {
```
In which it builds and sends a TCP packet
```
send_tcp_raw(sd, &my_inaddr, &dst_inaddr, MAGICPORT, oport, i, NULL, 0);
```
Listens for returned traffic using the packet sniffer (from tcpdump)
```
cp = read_pkt_sniffer(pd);
```
It sends another TCP packet
```
send_tcp_raw(sd, &my_inaddr, &dst_inaddr, MAGICPORT, cport, i, NULL, 0);
```
And again listens to the sniffer for a response
```
cp = read_pkt_sniffer(pd);
```
And, he processes then prints the tcp returned Flags
```
if (!fl) printf("(null) ");
if (fl & TH_URG)  printf("URG ");
if (fl & TH_ACK)  printf("ACK ");
if (fl & TH_PUSH) printf("PSH ");
if (fl & TH_RST)  printf("RST ");
if (fl & TH_SYN)  printf("SYN ");
if (fl & TH_FIN)  printf("FIN ");

if (cp) {
  memcpy(cresp, cp, 40);
  strip_ip_fields(cresp);
   /* hdump(cresp, 40); */
}
if ((op && !cp) || (cp && !op)) {
  printf("(dropped) ");
  /* might be ploss */
  printflags(i);
}

}
```

Conclusion

Serious hostile computer intrusion requires network scanning. These scans are designed to map networks, enumerate operating systems, identify software services, catagorize firewalls and firewall rules, and find out the OS version numbers where ever possible. Destination responses to TCP packets with malformed flags and reserved bits provide a rich set of information from which operating system finger printing is possible. There exist a wide variety of such tools for hostile use. Tools like tft.c, provide additional avenues for hostile tool extension.

Appendix: Source Code for tft.c

Editorial Comment: He seems to suffer from e. e. cummings envy ?

```
/* tft.c
Lamont Granquist
Tue Jul  7 23:30:36 PDT 1998 [ http://www.rootshell.com/ ]

"TCP Flag Test" -- 'excersizes' a machines TCP/IP stack by passing it all combinations of 64
TCP flags and seeing which flags are usable to determine which ports on the machine are open
or not.  it takes as arguments the machine to test, plus 2 ports -- one which you already know is
open and one which you already know is closed.  it may hose the inetd process on the machine
being tested.  it might even hose the machine itself, although i've never seen this behavior
myself.  if it crashes your machine don't blame me -- no warantees implied and such.  caveat
hacker.

to compile:

cc -o tft tft.c -lpcap

you must have the libpcap library, and it'll need to know where to find the pcap.h file (tip: this
means you'll need a -I flag most likely).  Your pcap version may need to be >= 0.4a6, this can be
difficult to determine, so if in doubt, just find a fresh copy to download. this works on NetBSD.  as
it uses SOCK_RAW, you must be r00t to run it -- although it doesn't bother to inform you of this
nicely, it'll just die with something like "socket: permission denied."  it will probably not work on
Solaris versions less than 2.6, or FreeBSD, or any other Unix which has funky byte ordering
problems in SOCK_RAW. i believe that SOCK_RAW tends to cause KERNEL PANICS in Digital
Unix, so don't try to run this on a DU box. this will probably not compile cleanly on unixes other
than NetBSD, and

I HAVE ABSOLUTELY NO INTENTION OF ATTEMPTING TO PORT OR SUPPORT THIS
PROGRAM ON DIFFERENT PLATFORMS.  I DON'T WANT ANY E-MAIL ABOUT THIS
PROGRAM.

if you have difficulties porting this script to another platform, I suggest you check out nmap
(www.insecure.org/nmap/index.html) and see how nmap is ported to your architecture, then hack
together something similar.  if nmap isn't ported to your architecture, then even if you dug up my
e-mail address i woulnd't be able to help you, so don't even bother trying then (and if nmap _is_
ported to your architecture then you need to do the work yourself, so don't bother then either...).
e-mail me with offers of sex or money *ONLY*.

any swiped code is probably fyodor's (apologies). this program could probably be built into
something that would attempt to 'fingerprint' a given TCP/IP stack and report back on the
probable type of O/S running on the machine -- see various papers on 'active probing' which
have been published in the security literature – Comer and Lin's paper is one example that i've
become familiar with.  this may have already been done by someone else in the lit, in which case
i offer my apologies, but the motivation for this was based on Uriel Maimon's suggestion in
Phrack P49-15 that other flags might be suitable for "FIN scanning".

if you don't understand the output, read the code, if you don't understand the code, give up.  this
program will not get you r00t access on anything and is not a (very good) denial-of-service
attack.  if you don't understand what this code does, do not lose sleep over it.
*/

ifdef HAVE_INLINE
#define __inline__  inline
#else
#define __inline__
```

```
#endif

#include "pcap.h"
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <malloc.h>
#include <math.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <arpa/inet.h>
#include <netdb.h>

#ifndef MAXHOSTNAMELEN
#define MAXHOSTNAMELEN 64
#endif

#define MAGICPORT 0xc23c

char            myname[MAXHOSTNAMELEN+1];
struct in_addr  my_inaddr;

__inline__
unsigned short in_cksum(unsigned short *ptr,int nbytes) {

  /* swiped -- see TCP/IP Illustrated */

  register long           sum;            /* assumes long >= 32 bits */
  u_short                 oddbyte;
  register u_short        answer;         /* assumes u_short == 16 bits */

  sum = 0;
  while (nbytes > 1)  {
    sum += *ptr++;
    nbytes -= 2;
  }

  if (nbytes == 1) {
    oddbyte = 0;
    *((u_char *) &oddbyte) = *(u_char *)ptr;
    sum += oddbyte;
  }

  sum  = (sum >> 16) + (sum & 0xffff);
  sum += (sum >> 16);
  answer = ~sum;
  return(answer);

}
```

```
void hdump(unsigned char *bp, int length) {

  /* stolen from tcpdump, then kludged extensively */

  static const char asciify[256] = "................................
!\"#$%&'()*+,-
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{
|}~.................................
!\"#$%&'()*+,-
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{
|}~.";

  register const u_short *sp;
  register const u_char *ap;
  register u_int i, j;
  register int nshorts, nshorts2;
  register int padding;

  printf("\n\t");
  padding = 0;
  sp = (u_short *)bp;
  ap = (u_char *)bp;
  nshorts = (u_int) length / sizeof(u_short);
  nshorts2 = (u_int) length / sizeof(u_short);
  i = 0;
  j = 0;
  while(1) {
    while (--nshorts >= 0) {
      printf(" %04x", ntohs(*sp));
      *sp++;
      if ((++i % 8) == 0)
        break;
    }
    if (nshorts < 0) {
      if ((length & 1) && (((i-1) % 8) != 0)) {
        printf(" %02x  ", *(u_char *)sp);
        padding++;
      }
      nshorts = (8 - (nshorts2 - nshorts));
      while(--nshorts >= 0) {
        printf("     ");
      }
      if (!padding) printf("     ");
    }
    printf("  ");

    while (--nshorts2 >= 0) {
      printf("%c%c", asciify[*ap], asciify[*(ap+1)]);
      ap += 2;
      if ((++j % 8) == 0) {
        printf("\n\t");
        break;
      }
    }
```

```
    if (nshorts2 < 0) {
      if ((length & 1) && (((j-1) % 8) != 0)) {
        printf("%c", asciify[*ap]);
      }
      break;
    }
  }
  if ((length & 1) && (((i-1) % 8) == 0)) {
    printf(" %02x", *(u_char *)sp);
    printf("                                %c", asciify[*ap]);
  }
  printf("\n");
}


__inline__
int send_ip_raw(int sd, char *packet, unsigned short datalen,
                unsigned char proto, struct in_addr *source,
                struct in_addr *dest) {

  /* all we do is add the ip header to the packet and send it */

  struct ip         *ip = (struct ip *) packet;
  struct sockaddr_in sock;
  int res;

  bzero((char *)packet, sizeof(struct ip));

  sock.sin_family      = AF_INET;
  sock.sin_port        = htons(20934);  /* does it matter? */
  sock.sin_addr.s_addr = dest->s_addr;

  ip->ip_v          = 4;
  ip->ip_hl         = 5;
  ip->ip_len        = htons(sizeof(struct ip) + datalen);
  ip->ip_id         = 0xc05e; /* rand(); */
  ip->ip_ttl        = 255;
  ip->ip_p          = proto;
  ip->ip_src.s_addr = source->s_addr;
  ip->ip_dst.s_addr = dest->s_addr;
  ip->ip_sum        = in_cksum((unsigned short *)ip, sizeof(struct ip));

  if ((res = sendto(sd, packet, ntohs(ip->ip_len), 0,
     (struct sockaddr *)&sock, (int) sizeof(struct sockaddr_in))) == -1) {
    perror("sendto in send_ip_raw");
    return -1;
  }

  return res;

}

int send_tcp_raw(int sd, struct in_addr *source, struct in_addr *dest,
                 unsigned short sport, unsigned short dport, unsigned char
```

```
                flags, char *data, unsigned short datalen) {

  /* sd should be SOCK_RAW + IP_HDRINCL */

  struct phdr {  /* tcp pseudo header */
    unsigned long s_addy;
    unsigned long d_addy;
    char zer0;
    unsigned char protocol;
    unsigned short length;
  };

  char          *packet = (char *) malloc(sizeof(struct ip)
                      + sizeof(struct tcphdr) + datalen);
  struct tcphdr *tcp    = (struct tcphdr *) (packet + sizeof(struct ip));
  struct phdr   *pseudo = (struct phdr *) (packet + sizeof(struct ip)
                      - sizeof(struct phdr));

  bzero((char *)packet, sizeof(struct ip) + sizeof(struct tcphdr));

  if (data)
    memcpy(packet + sizeof(struct ip) + sizeof(struct tcphdr), data, datalen);

  pseudo->s_addy   = source->s_addr;
  pseudo->d_addy   = dest->s_addr;
  pseudo->protocol = IPPROTO_TCP;
  pseudo->length   = htons(sizeof(struct tcphdr) + datalen);

  tcp->th_sport = htons(sport);
  tcp->th_dport = htons(dport);
  tcp->th_seq   = rand() + rand();
  tcp->th_ack   = rand() + rand();
  tcp->th_urp   = rand();
  tcp->th_off   = 5;
  tcp->th_flags = flags;
  tcp->th_win   = htons(2048);

  tcp->th_sum   = in_cksum((unsigned short *)pseudo, sizeof(struct tcphdr) +
                sizeof(struct phdr) + datalen);

  return(send_ip_raw(sd, packet, datalen+sizeof(struct tcphdr), IPPROTO_TCP,
        source, dest));

}

__inline__
void sethdrinclude(int sd) {
  int one = 1;
  setsockopt(sd, IPPROTO_IP, IP_HDRINCL, (void *) &one, sizeof(one));
}

void initialize(void) {
  struct hostent *myhostent;
```

```
  srand(time(NULL));

  if (gethostname(myname, MAXHOSTNAMELEN) == -1) {
    perror("gethostname");
    exit(1);
  }
  if ((myhostent = gethostbyname(myname)) == NULL) {
    perror("gethostbyname");
    exit(1);
  }

  memcpy(&my_inaddr, myhostent->h_addr_list[0], sizeof(struct in_addr));

}

__inline__
int resolve_host(char *host, struct in_addr *host_inaddr_p) {

  struct hostent *host_hent;

  if (inet_aton(host, host_inaddr_p) == 0) {
    if ((host_hent = gethostbyname(host)) != NULL) {
      memcpy(host_inaddr_p, host_hent->h_addr_list[0], sizeof(struct in_addr));
    } else {
      return(-1);
    }
  }
  return(0);
}

pcap_t *open_pkt_sniffer(char *cmdbuf, int snaplen, char *device) {

  bpf_u_int32         localnet, netmask;
  struct bpf_program  fcode;
  pcap_t              *pd;
  char                ebuf[PCAP_ERRBUF_SIZE];
  int                 i;

  if (device == NULL) {
    device = pcap_lookupdev(ebuf);
    if (device == NULL) {
      fprintf(stderr, "%s", ebuf);
      return(NULL);
    }
  }
  pd = pcap_open_live(device, snaplen, 0, 100, ebuf);
  if (pd == NULL) {
    fprintf(stderr, "%s", ebuf);
    return(NULL);
  }
  i = pcap_snapshot(pd);
  if (snaplen < i) {
    fprintf(stderr, "snaplen raised from %d to %d", snaplen, i);
    snaplen = i;
```

```
  }
  if (pcap_lookupnet(device, &localnet, &netmask, ebuf) < 0) {
    localnet = 0;
    netmask  = 0;
    fprintf(stderr, "%s", ebuf);
  }
  if (pcap_compile(pd, &fcode, cmdbuf, 1, netmask) < 0) {
    fprintf(stderr, "%s", pcap_geterr(pd));
    return(NULL);
  }
  if (pcap_setfilter(pd, &fcode) < 0) {
    fprintf(stderr, "%s", pcap_geterr(pd));
    return(NULL);
  }

  return(pd);

}

char *read_pkt_sniffer(pcap_t *pd) {
  static pcap_t      *last = NULL;
  struct pcap_pkthdr  head;
  static int          offset;
  char               *p;
  int                 datalink;
  int                 i = 0;

  if (!last || pd != last) {
    if ((datalink = pcap_datalink(pd)) < 0) {
      fprintf(stderr, "no datalink info: %s\n", pcap_geterr(pd));
      return(NULL);
    }
    switch(datalink) {
      case DLT_EN10MB:
        offset = 14; break;
      case DLT_NULL:
      case DLT_PPP:
        offset =  4; break;
      case DLT_SLIP:
        offset = 16; break;
      case DLT_RAW:
        offset =  0; break;
      case DLT_SLIP_BSDOS:
      case DLT_PPP_BSDOS:
        offset = 24; break;
      case DLT_ATM_RFC1483:
        offset =  8; break;
      case DLT_IEEE802:
        offset = 22; break;
      default:
        fprintf(stderr, "unknown datalink type (%d)", datalink);
        return(NULL);
    }
    last = pd;
```

```
  }
  while (i++ < 2) {  /* why twice? i don't know */
    p = (char *) pcap_next(pd, &head);
    if (p) {
      p += offset;
      break;
    }
  }
  return p;
}

__inline__
void strip_ip_fields(char *packet) {
  struct ip     *ip  = (struct ip *)packet;
  struct tcphdr *tcp = (struct tcphdr *)(packet + sizeof(struct ip));

  ip->ip_id  = 0;
  ip->ip_sum = 0;

  tcp->th_sum   = 0;
  tcp->th_sport = 0;
  if(tcp->th_ack) tcp->th_ack = 1;
  if(tcp->th_seq) tcp->th_seq = 1;
  if(tcp->th_urp) tcp->th_urp = 1;
}

char *ip_field[] = { "ip_vhl"  , "ip_tos"  , "ip_len"  , "ip_len"  ,
                     "ip_id"   , "ip_id"   , "ip_off"  , "ip_off"  ,
                     "ip_ttl"  , "ip_p"    , "ip_sum"  , "ip_sum"  ,
                     "ip_src"  , "ip_src"  , "ip_src"  , "ip_src"  ,
                     "ip_dst"  , "ip_dst"  , "ip_dst"  , "ip_dst"  ,
                     "th_sport", "th_sport", "th_dport", "th_dport",
                     "th_seq"  , "th_seq"  , "th_seq"  , "th_seq"  ,
                     "th_ack"  , "th_ack"  , "th_ack"  , "th_ack"  ,
                     "th_xoff" , "th_flags", "th_win"  , "th_win"  ,
                     "th_sum"  , "th_sum"  , "th_urp"  , "th_urp" };
__inline__
int pkt_compare(char *p1, char *p2, int n) {
  int i;
  int fl = 0;

  for(i=0;i<n;i++) {
    if(*p1++ != *p2++) {
      if (i < 41) {
        printf("%s ", ip_field[i]);
      } else {
        printf("%d ", i);
      }
      fl++;
    }
  }
  return(fl);
}
```

```
__inline__
void printflags(int fl) {
  if (!fl) printf("(null) ");
  if (fl & TH_URG)  printf("URG ");
  if (fl & TH_ACK)  printf("ACK ");
  if (fl & TH_PUSH) printf("PSH ");
  if (fl & TH_RST)  printf("RST ");
  if (fl & TH_SYN)  printf("SYN ");
  if (fl & TH_FIN)  printf("FIN ");
  printf("\n");
}

int main(int argc, char *argv[]) {
  struct in_addr dst_inaddr;
  int sd;
  int i;
  pcap_t *pd;
  unsigned short oport, cport;
  char cmdbuf[256], dstaddr[32];

  initialize();

  if (argc != 4) {
    fprintf(stderr,"usage: tft target open_port closed_port\n");
    exit(1);
  }

  oport = atoi(argv[2]);
  cport = atoi(argv[3]);

  if(resolve_host(argv[1], &dst_inaddr) < 0) {
    fprintf(stderr, "i don't know who %s is\n", argv[1]);
    exit(1);
  }

  if ((sd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0) {
    perror("socket");
    exit(1);
  }

  sethdrinclude(sd);

  sprintf(dstaddr, "%s", inet_ntoa(dst_inaddr));
  sprintf(cmdbuf, "tcp and src host %s and dst host %s and dst port %d",
        dstaddr, inet_ntoa(my_inaddr), MAGICPORT);

  if ((pd = open_pkt_sniffer(cmdbuf, 60, NULL)) == NULL) {
    fprintf(stderr, "can't open packet sniffer\n");
    exit(1);
  }

  for(i = 0; i<64; i++) {
    char oresp[40], cresp[40];
    char *op, *cp;
```

```
    /* printf("doing %d\n", i); */
    send_tcp_raw(sd, &my_inaddr, &dst_inaddr, MAGICPORT, oport, i, NULL, 0);
    usleep(1000);
    op = read_pkt_sniffer(pd);
    if (op) {
      memcpy(oresp, op, 40);
      strip_ip_fields(oresp);
      /* hdump(oresp, 40); */
    }
    send_tcp_raw(sd, &my_inaddr, &dst_inaddr, MAGICPORT, cport, i, NULL, 0);
    usleep(1000);
    cp = read_pkt_sniffer(pd);
    if (cp) {
      memcpy(cresp, cp, 40);
      strip_ip_fields(cresp);
      /* hdump(cresp, 40); */
    }
    if ((op && !cp) || (cp && !op)) {
      printf("(dropped) ");
      /* might be ploss */
      printflags(i);
      continue;
    }
    if (op && cp) {
      if (pkt_compare(oresp,cresp,((u_short *)cresp)[1])) {
        printflags(i);
      }
    }
  }

  return(0);

}
```

Bibliography

"Level Two Intrusion Detection in Depth", Course Manuals,Volumes 1 – 5, Northcutt, Kessler, Pomeranz, Irwin, Brenton, Novak, and Roesch, http://www.sans.org, (2001)

Network Intrusion Detection, An Analyst's Handbook, S. Northcutt, New Riders, (1999).

Intrusion Signatures and Analysis, S. Northcutt, M. Cooper, M Fearnow, and K. Frederick, New Riders, (2001).

Hacking Exposed: Network Security Secrets and Solutions, Second Edition, J. Scambray, S. McClure, G. Kurtz, Osborne/McGraw Hill, (2001).

TCP/IP Illustrated, Volume 1, WR Stevens, Addison-Wesley, (1994).

TCP/IP Illustrated, Volume 2, WR Stevens, Addison-Wesley, (1995).

ftp://ftp.freeBSD.org/pub/FreeBSD/ports/packages/security/strobe-1.06.tgz
http://packetstormsecurity.org/Exploit_Code_Archive/fping.tar.gz
http://packetstormsecurity.org/new-exploits/tft.c
http://rootshell.com/archive-j457nxiqi3gq59dv/199807/tft.c.html
http://wwdsilx.wwdsi.com
http://www.cert.org/incident_notes/IN-98.04.html
http://www.foundstone.com/rdlabs/termsofuse.php?filename=superscan.exe
http://www.hackingexposed.com/tools/tools.html
http://www.insecure.org/nmap
http://www.insecure.org/nmap/nmap-fingerprinting-article.html
http://www.ipswitch.com
http://www.kyuzz.org/antirez/hping.html
http://www.marko.net/cheops
http://www.netscantools.com/nstdownload.html
http://www.nmrc.org/files/snt
http://www.nmrc.org/files/sunix/icmpenum-1.1.1.tgz
http://www.nwpsw.com
http://www.research.umbc.edu/~andy

Assignment 3 – "Analyze This"

Introduction

The purpose of this assignment is to prepare a detailed analysis of 30 days of IDS logs. I have chosen to study data obtained from http://www.research.umbc.edu/~andy for the month of June 2001. These data were acquired through the use of the open source snort IDS software, (http://www.snort.org). The data are provided as 30 sets of three files for each day of the month. The three files for each day record the portscans as derived from the "preprocessor portscan" in snort, the alert headers as obtained from the rules sets that can be obtained and modified from snort.org, and a file of "out of spec" alerts that are enumerations of alerts whose TCP/IP packets exhibit parameters that are outside of RFC specifications for network traffic. The predominance of data in the oos files reflect data derived from crafted packets of one kind or another. The month of June has yielded over 400 MB of snort data for analysis. This data has been sanitized so that revealing elements of the IP addresses have been replaced with my.net.

Executive Summary

IDS logs derived from monitoring network traffic during the month of June 2001 shows a large amount of hostile traffic both to and from the MY.NET.* network.

While this is a lot of activity, it must be remembered that this is also a large university site. Academic networks are plagued by computer security problems related to their sheer size, their well-known public identity, and their extremely heterogeneous computer network structure. Academic networks are notorious as networks lacking in centralized configuration and purchasing authority, and an excess of users, perhaps on the verge of flunking out, with way to much time on their hands, and who are for the first time in their lives located in a truly high performance network environment. Many of these problems could be addressed by the aggressive enforcement of university wide "appropriate computer use" policy, with public ally enforced consequences for deviation. However, all universities experience many of the same security issues experienced by this site over the last month. Certainly, this site, which seems fairly well maintained, does not experience more than its share of intrusive activity.

The evidence presented in this report, suggest that there are computers within the university that are in need of investigation for signs of either compromise of improper use. Much of this activity derives from a relatively few number of subnets.

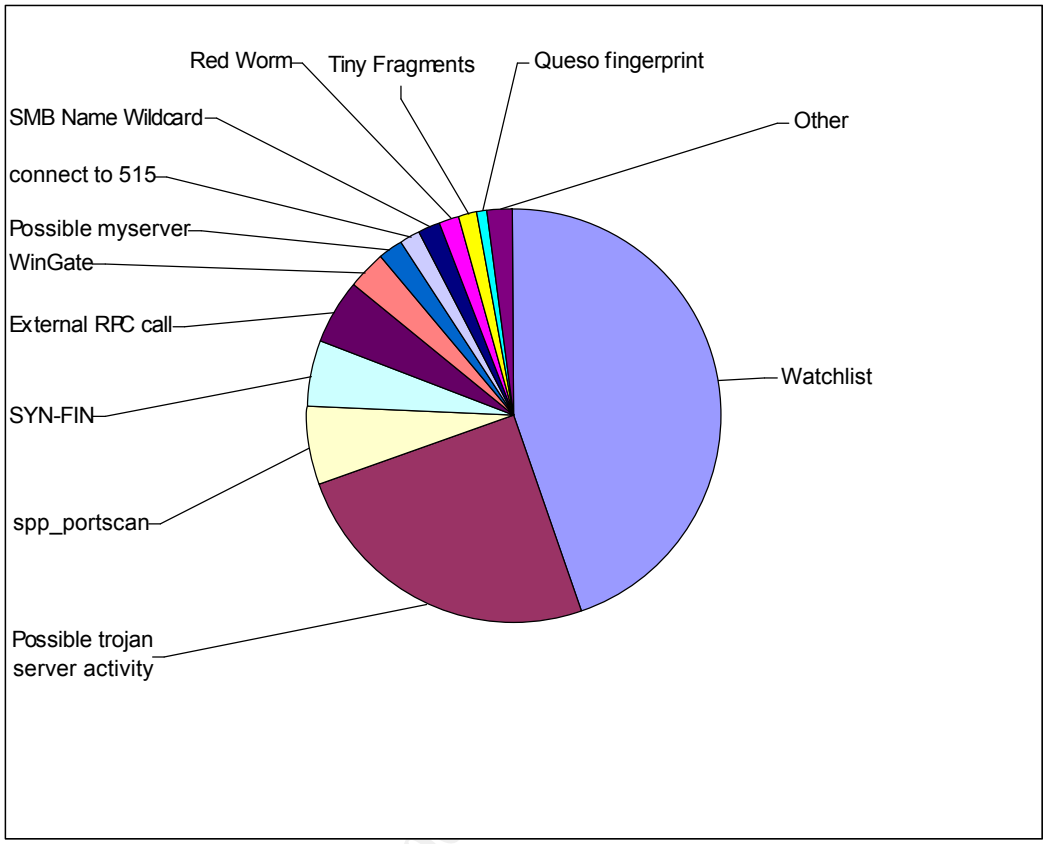
In addition, the university is a popular target for probes, system wide portscans, and directed assaults onto specific computers. All of this activity is essentially outside of the control of responsible figures in the university. However, the nature the external traffic suggests that there are several ports that the university might want to block at the firewall. In addition, there are some external subnets with a particularly bad record of hostile activity directed against MY.NET, that are good candidates for blocking at the university firewall.

Specific recommendations are presented in the “Defensive Recommendations” section.

Statistics

Over the period of 1 June -> 30 June 2001, the dataset exhibits the following alerts.

Alert	Count
Watchlist 000220 IL-ISDNNET-990517	153539
Possible trojan server activity	85286
spp_portscan: PORTSCAN DETECTED	20633
SYN-FIN scan!	18728
External RPC call	16291
WinGate 1080 Attempt	10685
Port 55850 tcp - Possible myserver activity - ref. 010313-1	6474
connect to 515	5943
SMB Name Wildcard	5705
High port 65535 tcp - possible Red Worm - traffic	5591
Tiny Fragments - Possible Hostile Activity	4764
Queso fingerprint	3132
High port 65535 udp - possible Red Worm - traffic	2311
Watchlist 000222 NET-NCFC	1760
Back Orifice	1283
TCP SRC and DST outside network	489
Null scan!	350
SUNRPC highport access!	214
NMAP TCP ping!	204
Attempted Sun RPC high port access	134
ICMP SRC and DST outside network	40
Russia Dynamo - SANS Flash 28-jul-00	26
STATDX UDP attack	6
SNMP public access	5
Probable NMAP fingerprint attempt	4
SITE EXEC - Possible wu-ftpd exploit - GIAC000623	2
hax0r boy 010615	1



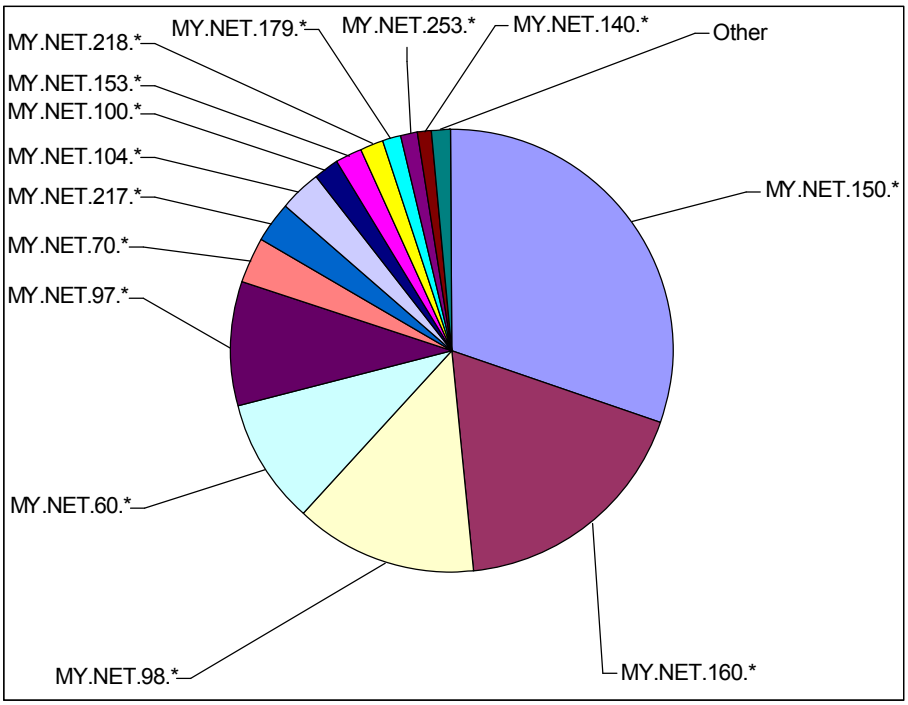
Alerts generated by the university IDS during the month of June 2001

Over the period of 1 June -> 30 June 2001, the dataset exhibits over a million portscans sourced from within the university. Fourteen of the total 51 subnets within the university are responsible for over 10,000 portscans each. Three networks are responsible for over 100,000 portscans. False positives derived from ntpd and within the university dns traffic have been pruned from these results.

Subnet	Source Portscan Count	Subnet	Source Portscan Count
MY.NET.150.*	327191	MY.NET.182.*	290
MY.NET.160.*	193825	MY.NET.201.*	206
MY.NET.98.*	142562	MY.NET.19.*	204
MY.NET.60.*	99759	MY.NET.137.*	188
MY.NET.97.*	96846	MY.NET.181.*	171
MY.NET.70.*	35580	MY.NET.184.*	111

MY.NET.217.*	33958
MY.NET.104.*	32450
MY.NET.100.*	21560
MY.NET.153.*	21177
MY.NET.218.*	17187
MY.NET.179.*	14397
MY.NET.253.*	12763
MY.NET.140.*	11479
MY.NET.69.*	3213
MY.NET.219.*	1747
MY.NET.6.*	1679
MY.NET.156.*	1627
MY.NET.71.*	1173
MY.NET.138.*	972
MY.NET.162.*	887
MY.NET.110.*	742
MY.NET.53.*	739
MY.NET.111.*	581
MY.NET.75.*	570
MY.NET.130.*	301

MY.NET.109.*	87
MY.NET.106.*	81
MY.NET.157.*	50
MY.NET.85.*	49
MY.NET.143.*	42
MY.NET.1.*	35
MY.NET.115.*	28
MY.NET.7.*	21
MY.NET.152.*	20
MY.NET.5.*	19
MY.NET.145.*	18
MY.NET.182.*	16
MY.NET.107.*	16
MY.NET.105.*	16
MY.NET.17.*	14
MY.NET.99.*	9
MY.NET.163.*	9
MY.NET.15.*	8
MY.NET.101.*	1



Internal subnets responsible for portscans during the month of June 2001

During the period of 1 June -> 30 June 2001, the dataset shows nearly 900,000 portscans sourced from outside the university directed within. This portscan traffic derives from 1561 IP addresses, 32 of which probed the university more than 10,000 times and 235 of which probed the university more than 100 times.

This table shows the 32 most active outside portscanners during June 2001. All these sites probed Universities more than 10,000 times each.

32 Most active Outside Hostile Port Scanners	IP name or Organization	Porscan Count	Target IP Address Count
193.253.243.190	APuteaux-102-1-5-190.abo.wanadoo.fr	43135	16983
205.188.233.153	g2lb5.spinner.com	39390	31
213.93.23.218	e23218.upc-e.chello.nl	37543	16718
205.188.233.121	g2lb4.spinner.com	36801	33
61.219.90.189	61-219-90-189.HINET-IP.hinet.net	29123	22811
198.247.29.18	Verio, Inc. (NET-VRIO-198-247)	28851	15908
205.188.233.185	g2lb6.spinner.com	26106	35
205.188.244.121	g2lb1.spinner.com	23898	29
205.188.246.121	America Online, Inc (NETBLK-AOL-DTC)	22848	29
213.56.40.58	ca-ol-montpellier-1-58.abo.wanadoo.fr	21193	14756
139.134.102.192	BDIP-T-010-p-102-192.tmns.net.au	20050	12829
217.81.194.157	pD951C29D.dip.t-dialin.net	19508	13976
203.34.37.133	dialin04.inverell.northnet.com.au	17117	12821
205.188.244.249	g2lb2.spinner.com	16539	28
211.184.223.2	DEOKSAN ELEMENTARY SCHOOL, KR	16226	11258
128.32.131.127	erlik.CS.Berkeley.EDU	16124	16124
217.136.37.76	adsl-42316.turboline.skynet.be	15265	10321
195.190.34.55	main.texnikoi.gr	14451	11004
211.240.28.66	ITBUSINESS, KR	14378	14319
217.58.147.39	Unknown	13940	9814
217.75.226.210	dns.dammedia.es	13894	13567
217.80.206.28	pD950CE1C.dip.t-dialin.net	13735	10201
210.125.151.139	CHUNGBUK NATIONAL UNIVERSITY, KR	13369	10182
211.72.171.75	Grand Tek Technology Co., Ltd., TW	13276	10778
200.181.53.131	Brasil Telecom S.A., BR	12971	12921
210.47.244.15	Dalian Medical University, CN	12648	10319
165.230.53.35	conklina25.rutgers.edu	12575	11526
194.100.55.131	mars.tvk.fi	12368	9582
213.100.81.113	catv-213-100-81-113.swipnet.se	11370	7732
212.179.225.193	bzq-225-193.bezeqint.net	11095	9253
207.236.81.82	Dolisos Canada Inc., CA	10867	5566
192.204.190.76	Verio, Inc. (NET-VRIO-192-204)	10637	10632

Editorial Aside: I am fascinated by the following observation. These data come from an important American university. I am a system/security administrator in an academic

research environment, but wholly unrelated to this one. The addresses observed in this "Analyze This" assignment are many of the same addresses that I encounter probing my network as well. See quote by Benjamin Franklin in "Detect 4".

These data reveal two kinds of portscans. All of these portscans involve over 10,000 sessions to various computers on the university net. But, some of the portscans (c.f APuteaux-102-1-5-190.abo.wanadoo.fr) are scans to over 10,000 IP addresses. These are obviously portscans that targeted most of the computers in the entire university looking for one or a very few open ports. On the other hand 6 of these scans are directed at fewer than 36 computers (c.f . *.spinner.com, and there are 5 such scans). That is to say, these 6 scans are deeply focused on just a few computers and each of those computers is probed many times.

This table shows the 6 hostile IP addresses that were focused on a few local IP addresses only. Additionally, it shows the top ten targets of each for those hostile IP addreses.

Hostile IP Address	Targets This Many IPs	Top Ten Local Targets	Portscan Count
205.188.233.153	31	MY.NET.108.15	3901
		MY.NET.70.92	3635
		MY.NET.110.33	3406
		MY.NET.178.154	3074
		MY.NET.145.197	2844
		MY.NET.107.4	2822
		MY.NET.110.169	2439
		MY.NET.108.13	2324
		MY.NET.178.222	1896
		MY.NET.145.166	1773
205.188.233.121	33	MY.NET.106.178	3114
		MY.NET.178.154	3080
		MY.NET.109.62	3062
		MY.NET.110.33	3014
		MY.NET.104.127	2461
		MY.NET.108.13	2224
		MY.NET.108.15	2179
		MY.NET.107.4	1932
		MY.NET.110.169	1862
		MY.NET.15.223	1769
205.188.233.185	36	MY.NET.145.166	2463
		MY.NET.106.178	2270
		MY.NET.15.223	1726
		MY.NET.110.33	1671
		MY.NET.107.4	1646

		MY.NET.110.169	1445
		MY.NET.108.15	1420
		MY.NET.178.188	1231
		MY.NET.70.92	1220
		MY.NET.178.222	1187
205.188.244.121	29	MY.NET.110.169	2332
		MY.NET.108.13	2323
		MY.NET.145.166	2063
		MY.NET.111.30	1944
		MY.NET.106.178	1664
		MY.NET.110.33	1657
		MY.NET.178.222	1604
		MY.NET.109.62	1485
		MY.NET.178.154	1187
		MY.NET.180.76	1073
205.188.246.121	29	MY.NET.178.154	2208
		MY.NET.108.13	1922
		MY.NET.145.166	1663
		MY.NET.110.169	1536
		MY.NET.70.92	1460
		MY.NET.145.197	1412
		MY.NET.106.178	1329
		MY.NET.178.222	1285
		MY.NET.107.4	1071
		MY.NET.110.33	1065
205.188.244.249	28	MY.NET.178.154	1863
		MY.NET.70.92	1623
		MY.NET.110.169	1353
		MY.NET.110.33	1271
		MY.NET.145.197	1013
		MY.NET.108.13	1001
		MY.NET.146.17	953
		MY.NET.106.178	947
		MY.NET.178.222	884
		MY.NET.108.15	772

Another interesting observation is presented in the next table. The 6 most focused hostile IP addresses were busy scanning the same local addresses. This table show how may times the top ten targets of the 6 most focused hostile IP addresses were targeted.

Target IP Address	Incident Count

MY.NET.110.169	6
MY.NET.110.33	6
MY.NET.106.178	5
MY.NET.108.13	5
MY.NET.178.154	5
MY.NET.178.222	5
MY.NET.107.4	4
MY.NET.108.15	4
MY.NET.145.166	4
MY.NET.70.92	4
MY.NET.145.197	3
MY.NET.109.62	2
MY.NET.15.223	2

The following table enumerates the ports that were communicated with in the month of June 2001. Ports appear in this table, if they were targeted more than 1000 times during the month.

Count	Port	Count	Port	Count	Port
2041167	5779	8948	1524	1541	7788
434232	21	7651	27500	1482	8889
292779	28800	6424	7003	1418	113
164497	53	5854	515	1347	7777
164011	6970	5822	47017	1280	27019
127268	1234	4489	6347	1271	41003
115365	27005	4368	9705	1070	28000
100492	27374	4352	55850	1070	27961
63384	6112	3366	27020	1053	21439
54377	23	3357	80	1045	22952
44709	1214	3262	4236	1041	24979
40196	7778	3179	27243	1039	7782
39952	6346	3035	27025	1033	27040
39439	9001	2910	44444	1013	27050
34933	137	2907	4020		
30633	111	2696	27960		
28836	24452	2247	27018		
19379	110	2227	2049		
19163	25	2100	7001		
18299	1080	1964	27035		
14385	4241	1800	2072		
11556	1033	1762	27045		
10294	31337	1726	27030		
9948	13139	1679	1025		
9614	7000	1618	2213		

The following table is an enumeration of the important target port traffic for the entire network during the month of June 2001. Lines in bold are important network ports used for network traffic (e.g. port 21, ftp). Some of the traffic directed against those ports likely reflect false positives; however, the remaining alerts represent hostile activity directed against the most crucial computational resources of the university. In addition, counts are enumerated for ports of associated with known hostile traffic (e.g SubSeven on port 27374), and finally counts are enumerated for all ports for which the traffic exceeds 10,000 episodes.

Port	Count	Port Description
0	217	Reserved
20	**147**	**File Transfer [Default Data],**
21	**434232**	**File Transfer [Control],**
22	**218**	**SSH Remote Login Protocol, SSH Remote Login Protocol**
23	**54377**	**Telnet,**
25	**19163**	**Simple Mail Transfer, mail**
53	**164497**	**Domain Name Server,**
67	**163**	**bootps**
69	**118**	**Trivial File Transfer,**
79	**105**	**Finger,**
80	**3357**	**World Wide Web HTTP, World-Wide-Web protocol**
110	**19379**	**POP version 3**
111	**30633**	**sunrpc SUN Remote Procedure Call**
137	**34933**	**NETBIOS Name Service, NetBIOS Name Service**
138	**157**	**NETBIOS Datagram Service, NetBIOS Datagram Service**
139	**107**	**NETBIOS Session Service, NetBIOS Datagram Service**
143	**378**	**Imap**
152	104	Background File Transfer Program
161	**106**	**SNMP, Simple Net Mgmt Proto**
389	**102**	**Lightweight Directory Access Protocol**
481	**105**	**Ph service**
529	101	IRC-SERV
530	101	rpc
1001	104	Der Späher / Der Spaeher, Le Guardien, Silencer, WebEx
1033	11556	Unknown Port
1080	18299	Socks, socks proxy server
1214	44709	KAZAA
1234	127268	SubSeven, Infoseek Search Agent
1243	147	SubSeven
2140	28	Deep Throat
4241	14385	VRML Multi User Systems
5779	2041167	Unknown Port
6000	**30**	**X-server**

6112	63384	dtspcd, CDE subprocess control
6346	39952	gnutella-svc
6347	4489	gnutella-rtr
6970	164011	GateCrasher
7778	40196	Interwise
9001	39439	Unknown Port
12345	898	NetBus (and many variants)
12346	22	NetBus
24452	28836	Unknown Port
27374	100492	SubSeven, SubSeven variants, Ramen
28800	292779	Unknown Port
31337	10294	Hi Back Orifice trojan horse
44444	2910	Trojan: Prosiak

Detects by Priority with Descriptions

Presented below are brief descriptions for each of the observed alerts obtained during the month of June 2001. Within most of the descriptions enumerated below are short tables of data. Unless otherwise noted, these tables contain two columns. The first column lists the number of counted detects as described in the paragraph, and the second column lists the IP address, subnet, or port number associated with the counts. If there is a third column, it will be there to make a comment or recommendation.

Attempted Sun RPC high port access

Five external cpmputers attempted to communicate with Solaris workstations on the high-number RPC ports. The two most serious offenders were:

113	205.188.153.101
10	205.188.153.103

They targeted 6 MY.NET computers, especially these:

99	MY.NET.217.18	Investigate for compromise
14	MY.NET.217.38	Investigate for compromise
10	MY.NET.97.237	Investigate for compromise

These communications were sourced on port 4000 and 53, and were targeted to port 32771 (portmapper) on the targets. This is an attempt to look for misconfigured and vulnerable services in Solaris Unix boxes.

Back Orifice

There were 16 off campus sites that attempted communication with MY.NET computers through the Back Orifice port. The most serious were:

231	203.107.244.195
180	203.155.244.91
168	203.146.127.236
128	203.107.244.13

There were 254 targets on MY.NET for this activity. No computer in MY.NET received more than 10 scans looking for Back Orifice; these were scans looking for BO. Most of these attempts look like searches with no positive detects.

connect to 515

There were 22 external IP addresses that attempted to connect with printers on the MY.NET using the LPD printer service on port 515. Some of this activity occurred hundreds of time per external address. The top five offenders were:

1422	64.27.27.1	
774	150.183.110.179	
622	202.109.72.113	
450	216.139.196.151	
308	161.184.162.126	

These were university wide scans, that did not focus on a few subnets of workstations. There are worms that exploit the lpd deamon, see http://www.sans.org/y2k/040401.htm. It would seem reasonable for the university to consider blocking access to port 515 from the outside, in order to stop theses scans.

External RPC call

There were 66 external computers that made RPC calls to on campus computers. The 10 worst offenders were:

1304	202.98.10.70	
1243	61.143.127.86	
1229	134.198.26.42	
1176	211.152.241.1	
800	129.49.65.82	
759	212.209.79.162	
734	24.147.14.159	
651	128.95.12.195	
614	129.186.213.89	
394	24.27.62.134	

1279 MY.NET.* computers were targeted in this activity. The activity was not focused on a few networks or workstations, rather it was spread throughout the university. There seem to have bee some 17 scans that touched large parts of the network. Of course, ll of this traffic was directed at port 111. and was motivated by search for vulnerable rpc services (especially on SUN workstations) that might be subject to buffer overflow and nfs exported file systems accessible by the world.

hax0r boy 010615

On one occasion the onsite computer MY.NET.60.11:23, attempted an attack on 24.19.166.5.
This is trojan activity.

High port 65535 tcp - possible Red Worm – traffic

There were 72 computers, 22 of which were internal, that attempted to communicate over port 65535. This port has been associated with Red Worm trojan activity (http://www.datafellows.com/v-descs/adore.shtml). The worst offenders were:

4918	192.207.123.2	
178	MY.NET.253.24	Investigate for compromise

67	64.12.168.249	
52	205.188.156.154	
24	MY.NET.6.47	Investigate for compromise
20	MY.NET.6.34	Investigate for compromise

The primary targets were:

4918	MY.NET.99.51	Investigate for compromise
103	MY.NET.253.24	Investigate for compromise
67	MY.NET.111.139	Investigate for compromise
62	205.188.156.154	
60	199.154.149.191	
22	195.121.6.51	

As part of it's activity the Red Worm/Adore installs a backdoor, that when activated by a specific ping packet, will open a back door root shell that listens on port 65535. This scan is looking for those ports. The targets of this activity should be inspected for evidence of this compromise.

High port 65535 udp - possible Red Worm – traffic

There are 2311 alerts concerning traffic form or to port 65535, which has lately been associated with the Red Worm trojan. Most of this traffic derived from the following sources:

2064	216.169.36.189	
30	217.59.83.44	
26	MY.NET.70.242	Investigate for compromise
23	195.200.18.28	
22	217.59.83.45	
13	64.182.96.150	
10	212.27.54.28	

And most of the traffic has been directed against

2118	MY.NET.70.242	Investigate for compromise
52	MY.NET.163.54	Investigate for compromise
31	MY.NET.160.169	Investigate for compromise
16	MY.NET.69.209	Investigate for compromise
11	64.40.88.100	

This traffic has predominately been associated with these ports:

From Ports:		To Ports:	
Count	**Port**	**Count**	**Port**
2192	65535	2107	27960
52	5314	119	65535
26	27961	23	27961
22	6112	12	53

Particular attention should be paid to

216.169.36.189:65535 -> MY.NET.70.242:27960	2064 alerts
217.59.83.45:5314 -> MY.NET.163.54:65535	22 alerts
217.59.83.44:5314 -> MY.NET.163.54:65535	30 alerts

Significant traffic has gone in both directions to these MY.NET computers. It is almost certain that they have been compromised with Red Worm.

ICMP SRC and DST outside network

There were 40 detects of ICMP packets crossing the IDS with source and destination addresses outside MY.NET.*. This is evidence of packet crafting. It indicates that there are either compromised workstation(s) or uncompromised workstations operated by people up to no good. At very least the source addresses are spoofed. It will be difficult to track down these computers. It might be useful to watch for ingress activity at the firewall from the computers targeted In these alerts. If the targets return to the university, their future targets my provide a clue or starting place to look for the internal packet crafters.

See below in the alert "TCP SRC and DST outside network". These episodes are similar, except for use of TCP packets. None of the destination IP address in these incidents are the same. From the time stamps, it is clear that while some of the activity on the two workstations occurred on the same day; in only a couple of cases do the time of day stamps overlap when the activity occurs on the same day. These two alerts likely reflect activity from an unrelated but similarly compromised workstations in MY.NET.

NMAP TCP ping!

Thirty four exterior sites sent NMAP generated TCP pings to 34 MY.NET computers. These packets came predominately from:

70	209.135.37.205
22	207.238.101.253
21	202.187.24.3
19	204.167.220.253
16	199.197.130.21

And were targeted mostly at:

72	MY.NET.1.8
14	MY.NET.1.3
13	MY.NET.60.14
12	MY.NET.253.125
12	MY.NET.1.9
12	MY.NET.100.165

The TCP ping is a tool use to port scan and perform operating system finger printing.

Null scan!

There were 234 null scans directed against MY.NET. The heaviest traffic came from:

19	62.252.40.153

10	62.243.160.209
9	202.92.71.208
7	24.7.213.142
7	24.29.186.167
6	24.79.67.190
5	24.226.169.228

The most common targets were:

88	MY.NET.70.97
37	MY.NET.217.18
29	MY.NET.150.133
28	MY.NET.70.66
15	MY.NET.150.220
10	MY.NET.70.77
10	MY.NET.219.50
10	MY.NET.218.126
10	MY.NET.150.225

A null scan is a technique whereby TCP packets with no flags are sent to target computers, in hopes of OS fingerprinting and network reconnaissance. These packets are set with sequence number equal to 0. This is crafted traffic, that does not normally occur. Such activity is usually a precursor to a more serious attack.

Port 55850 tcp - Possible myserver activity - ref. 010313-1

This activity derives form 78 sources within and offsite from the University. However the prime offenders seem to be:

4144	MY.NET.1.6	talking to	128.8.128.180
1785	64.213.55.2	talking to	MY.NET.130.122

This is trojan activity.
http://www.sans.org/y2k/082200.htm

Possible trojan server activity

There are 3569 alerts associated with trojan activity. The serious hostiles are:

31890	MY.NET.70.38	Investigate for compromise
11073	216.220.167.76	
7050	129.170.104.19	
6006	MY.NET.146.95	Investigate for compromise
5606	216.220.164.141	Investigate for compromise
2440	205.157.65.4	
1293	204.210.139.127	

And the most common targets are:

7906	216.220.167.76	
6006	205.157.65.4	
4675	216.220.164.141	
4454	MY.NET.218.82	Investigate for compromise
4144	128.8.128.180	
2738	129.170.104.19	

2440	MY.NET.146.95	Investigate for compromise
1786	MY.NET.130.122	Investigate for compromise

The target ports most commonly communicated with are 27374 (SubSeven), 55850 (unknown), 6346 (Gneutella), 2072 (inknown). The MY.Net computers need to be investigated for possible signs of trojan compromise.

Probable NMAP fingerprint attempt

These off site computers performed a fingerprint probe on MY.NET.

1 24.201.107.143
1 151.112.2.25

This probe targeted:

1 MY.NET.60.8
1 MY.NET.218.22

Additionally the computer MY.NET.100.65:62178, performed a similar fingerprint against MY.NET.101.141:7 on 2 occasions. The NMAP fingerprint probe uses TCP packets with improper flag settings. Different operating systems will respond in various ways to such traffic. Reconnaissance techniques such as this, often precede more serious attacks.

Queso fingerprint

Russia Dynamo - SANS Flash 28-jul-00

The Russian dynamo is a windows trojan that gathers windows configuration information and sends it to 194.87.6.255 (http://www.sans.org/y2k/072818.htm). It would be prudent to block this class C address range at the university firewall. The MY.NET computers that have been communicating with this Russian subnet, and therefore have likely had configuration and passwd information compromised are the following:

7	MY.NET.182.120	Investigate for compromise
3	MY.NET.104.111	Investigate for compromise
2	MY.NET.70.97	Investigate for compromise

SITE EXEC - Possible wu-ftpd exploit - GIAC000623

The offsite computer 211.235.241.145 attempted to compromise the onsite computer MY.NET.144.59 on 2 occasions. The ftp server, wu-ftp, has a known vulnerability to root compromise when used in the "site exec" mode. This computer need to be checked to insure that the ftp service has not bee compromised (http://www.sans.org/y2k/063000.htm).

SMB Name Wildcard

There are 1125 external computers that attempted to enumerate open shares on a SAMBA server running in a Unix system. The probe, which is directed against port 137, is also used by windows NETbios name service, and can be used to enumerate open shares on windows systems. The worst offenders were:

1492	165.230.53.35
257	216.63.216.27

166	216.61.41.249
100	216.67.164.34

This activity manifested itself as scans on the entire MY.NET, and did not focus on a few subnets or workstations. The university should strongly encourage users to impose strong passwd on windows resources, as well as limiting the extent to which computational resources can be read. Through university policy, users should be strongly encouraged to insure good passwds on openly shared resources, especially Windows shared directories, and Unix file systems should be exported only to well delineated subnets within the university. If possible these should be checked by regular university scanning.

SNMP public access

This alert derives from an attempt by a source to administer or configure another computer using the SNMP protocol. The usual course of action is for the hostile computer to exploit the default passwd "public". The outside IP address 146.242.123.29 initiated SNMP traffic with MY.NET.134.1. This should be investigated to insure passwd integrity, to insure workstation integrity, and to disable the service if it is unused.

spp_portscan: PORTSCAN DETECTED

The university suffered 20633 portscans in the month of June. The statistics of these portscans are presented in the previous section.

STATDX UDP attack

This alert reflects an attempt to exploit the Red Hat rpc.statd service (http://www.sans.org/y2k/120600-1200.htm, http://www.kulua.org/Archives/kulua-l/200008/msg00159.html, http://www.whitehats.com/info/IDS442). The computer MY.NET.6.15 was assaulted on port 32776, by these off site computers:

2	212.209.79.162
1	210.90.168.5
1	210.107.198.164
1	139.142.135.118
1	129.49.65.82

This computer needs to be inspected to insure that it has not been compromised.

SUNRPC highport access!

There were 21 off campus sites that attempted to communicate with the Solaris RPC portmapper port in order to enumerate the high port number services enabled on Sun boxes. The worst offenders were:

45	129.244.36.81
31	66.26.252.85
19	35.9.37.225
16	66.26.255.103

And, the most common targets were port 32771 on:

45	MY.NET.218.78
44	MY.NET.217.198
26	MY.NET.217.18
24	MY.NET.218.146
19	MY.NET.100.153
15	MY.NET.253.52
10	MY.NET.6.7
10	MY.NET.253.51
10	MY.NET.179.78

SYN-FIN scan!

There were over 18,000 SYN-FIN scans directed against the network. All but 4 of them were directed against port 21 (ftp). The remaining 4 were directed against high ports (probably looking for trojans) and port 109 (pop2). These crafted packets are designed to find ftp servers with writable directories, to elicit information from the target, gather banners, to aid in targeting services for buffer overflow and in OS determination. There were 3 serious sources for the SYN-FIN probes:

14348	211.240.28.66
4220	61.13.106.35
156	211.114.44.2

These scans did not target specific workstations or subnets, they spanned the entire university computer system.

TCP SRC and DST outside network

There were 489 detects of TCP packets crossing the IDS with source and destination addresses outside MY.NET.*. This is evidence of packet crafting. It indicates that there is either a compromised workstation(s) or uncompromised workstations operated by people up to no good. At very least the source addresses are spoofed.

Looking at the target ports extracted from these detects, suggests port scanning activity. All the detects organize into no more than 3 portscans, in addition these specific ports were communicated with more than 3 times:

80	21	ftp
59	0	Clear evidence of packet crafting
56	5190	AOL
19	8888	Unknown, but associated with special commercial services
10	1863	Unknown, but associated with special commercial services
6	3090	Unknown, but associated with special commercial services
5	1410	Unknown, but associated with special commercial services
4	3088	Unknown, but associated with special commercial services
4	119	news

There is no obvious evidence in this to suggest a possible method of compromise. However, looking at the source ports:

113	27374	SubSeven

59	0	Packet crafting
18	6346	Gnutella
18	2006	Unknown
14	1055	Unknown
13	111	Portmapper
12	53	DNS

This is a computer running or compromised with SubSeven. The activity has been going on all month. This computer(s) need to be found and cleaned.

Watchlist 000220 IL-ISDNNET-990517

There were in excess of 155,000 detects from the 156.226.0.0. and 212.179.0.0 class B networks. This activity derives especially from 212.179.58.200, 212.179.79.2, 212.179.56.5, 212.179.47.70, and 212.179.72.226. These attacks were directed against many subnets within the university, however primarily MY.NET.100.*, MY.NET.150.*, MY.NET.217.*, MY.NET.218.*, MY.NET.70.*, MY.NET.97.*, and MY.NET.98.*. Particular attention should be paid to the computers MY.NET.150.220, MY.NET.218.198, MY.NET.97.175, and MY.NET.97.210; these computers suffered nearly 150,000 contacts with these hostile sources. This activity is associated 62 ports, but especially with ports 1234, 4241, 4236, 4020, 1214, and 41003. One associates port 1243 with SubSeven, but SubSeven is configurable, and it is commonly seen on 1234. This activity likely reflects that several of the computers targeted are compromised with SubSeven. And the large amount of communication clearly indicated that many of these computers are compromised with some trojan.

Handlers at the Global Incident Analysis Center in their daily detects site, explore data quite similar to this, and have discussed the prospect that this traffic may be related to some "good citizen" grey hat who has built a worm that invades PCs through open shares but only leaves messages of an "instructional" nature. If you believe this interpretation (and it is the hostile's interpretation), then you can imagine it to be more benign than not. I would not treat it that way, and it does not look like the "handlers on duty" do either. It is a worm, it has no business on your net.
http://www.sans.org/y2k/051900.htm.
http://www.sans.org/y2k/052000.htm

These computers need special attention.

127165 MY.NET.150.220
14369 MY.NET.218.198
2987 MY.NET.97.44
2893 MY.NET.97.175
1268 MY.NET.97.210

Watchlist 000222 NET-NCFC

There were 21 computers on the 159.226.0.0 Class B net, that communicated with MY.NET during the month. These computers were especially busy:

681 159.226.45.3

607	159.226.41.166
137	159.226.121.37
59	159.226.39.26
53	159.226.5.94

This traffic was directed at 19 computers in MY.NET, especially

516	MY.NET.253.42
300	MY.NET.100.56
223	MY.NET.100.83
206	MY.NET.253.43
177	MY.NET.6.7

These addresses are associated with the recent heavy activity from the China. The activity communicated over the mail port as well as these high number ports 42513, 34164, 8765, 61490, 37027, 1431, 1523, and 1523 on MY.NET targets.

WinGate 1080 Attempt

There were 227 outside sites that attempted WinGate access to MY.NET computers. These were attempts to communicate with the WinGate Proxy Server. This service, which occurs on port 1080, can be used to hide the original source from further web surfing, or intrusive activity by making subsequent computer connection anonymous. Most of this activity was scan activity looking for available servers. These were the top 5 scanners:

693	208.151.245.252
584	24.200.15.30
527	24.249.236.109
504	24.130.201.49
476	62.54.255.94

The scans were not focused on a few networks or workstations, rather they were dispersed over the entire university network.

"Top Talkers" Analysis and Link Maps

During the month of June 2001, there were 9,309,042 individual incidents of portscan, alerts or oos packets. Including both sources and destinations, this traffic involves175,991 distinct IP addresses. The "Top Talkers" in this traffic defined as those with more than 40,000 individual episodes in either portscan, alert or oos logs are the following:

Incident Count	IP Address	Net Name or Organization
969712	63.250.213.73	Yahoo! Broadcast Services, Inc.
969712	233.28.65.227	Internet Assigned Numbers Authority (Reserved IP address)
873324	233.28.65.62	Internet Assigned Numbers Authority (Reserved IP address)
487253	63.250.213.119	Yahoo! Broadcast Services, Inc.
386071	63.250.213.124	kfogw.broadcast.com (Yahoo)
291209	MY.NET.160.114	MY.NET
155867	MY.NET.150.220	MY.NET

155318	MY.NET.150.133	MY.NET
140300	63.250.213.26	Yahoo! Broadcast Services, Inc.
140300	233.28.65.164	Internet Assigned Numbers Authority (Reserved IP address)
140184	MY.NET.150.225	MY.NET
127124	212.179.58.200	NV-PICTUREVISION, Bezeq International
109689	MY.NET.60.16	MY.NET
75475	MY.NET.70.38	MY.NET
51932	205.188.233.153	g2lb5.spinner.com
48201	205.188.233.121	g2lb4.spinner.com
45876	211.240.28.66	ITBUSINESS, KR
45007	193.253.243.190	APuteaux-102-1-5-190.abo.wanadoo.fr
44554	MY.NET.150.204	MY.NET

All of these IP addresses are associated with portscans, alerts or oos packets. Seven of the IP addresses are internal to the university. Two of the IP addresses are reserved addresses, and hence derive from packet crafting.

63.250.213.73 and 233.28.65.227
63.250.213.119 and 233.28.65.62
63.250.213.124 and 233.28.65.62
63.250.213.26 and 233.28.65.164

```
#------------------------------------------------------------------
#---- Monthly Communication Stats for 63.250.213.73
#------------------------------------------------------------------

#--- There were 1 src address and 1 src port involved with 63.250.213.73

#--- Of which the top ten source addresses were From:
 969712  63.250.213.73
#--- The top ten source ports were:
 969712  1042

#--- There was 1 des addresses and 1 des port involved with 63.250.213.73

#--- Of which the top ten destination addresses were To:
 969712  233.28.65.227
#--- The top ten destination ports were:
 969712  5779

#------------------------------------------------------------------
#---- Monthly Communication Stats for 233.28.65.62
#------------------------------------------------------------------

#--- There were 2 src addresses and 2 src ports involved with 233.28.65.62

#--- Of which the top ten source addresses were From:
 487253  63.250.213.119
 386071  63.250.213.124
#--- The top ten source ports were:
 487253  1036
```

```
 386071  1031

#--- There were 1 des addresses and 1 des ports involved with 233.28.65.62

#--- Of which the top ten destination addresses were To:
 873324  233.28.65.62
#--- The top ten destination ports were:
 873324  5779

#------------------------------------------------------------------
#---- Monthly Communication Stats for 63.250.213.26
#------------------------------------------------------------------

#--- There were 1 src addresses and 2 src ports involved with 63.250.213.26

#--- Of which the top ten source addresses were From:
 140300  63.250.213.26
#--- The top ten source ports were:
 126586  1038
  13714  1039

#--- There were 1 des addresses and 1 des ports involved with 63.250.213.26

#--- Of which the top ten destination addresses were To:
 140300  233.28.65.164
#--- The top ten destination ports were:
 140300  5779
```

This represents nearly one million packets between 2 IP addresses and just 2 ports; it must be a denial of service attack. The alert messagefor this traffic is "[**] UDP SRC and DST outside network". The destination address is a reserved IP address unavailable to the network; so, the destination must be crafted. There can be no return traffic to these crafted source IP addresses. The source address is from Yahoo. Packets that really are from Yahoo to a reserved destination address would not have been seen by the MY.NET firewall. Therefore the packets must have originated from within MY.NET. Therefore, the source address (Yahoo) must also be spoofed. This is a compromised MY.NET computer doing a DOS against the MY.NET firewall, or It is an on campus hostile in need of some administrative discipline.

Because the packets were found at the MY.NET firewall, then this was a denial of service attack from a reserved (crafted IP address) 63.250.213.73, directed against 233.28.65.227. This means that the traffic derives from inside the MY.NET, and has a spoofed source address and was targeting a yahoo computer.

The five noisiest talkers are undertaking this activity. One pair of IP addresses lasted all month. A second pair persisted for the first half of the month, and shifted to another pair of IP addresses for the second half. Because all the IP addresses are crafted in these three incidents, it is unknown if the this all derives from the same internal MY.NET computer of more than one. See the link maps below.

Denial of Service against MY.NET IDS Link Maps

The computer MY.NET.160.114 needs to be tracked down and looked at. It is port scanning predominantly from source port 777 has performed 28,220 portscans and 7726 IP target IP addresses against off site computers. It has either been compromised, or being used by an on campus hacker.

```
#-------------------------------------------------------------------
#---- Monthly Communication Stats for MY.NET.160.114
#-------------------------------------------------------------------

#--- There were 23 src addresses and 25 src ports involved with MY.NET.160.114

#--- Of which the top ten source addresses were From:
 189095  MY.NET.160.114
      2  217.136.37.76
      2  213.93.23.218
      2  165.230.53.35
      2  129.170.104.19
      1  62.243.115.13
      1  61.219.90.189
      1  217.81.194.157
      1  217.58.147.39
      1  217.57.19.30
#--- The top ten source ports were:
 189090  777
      2  4676
      2  3563
      2  2702
      2  2152
      2  21
      2  1623
      2  110
      1  4741
      1  4717

#--- There were 28220 des addresses and 7726 des ports involved with MY.NET.160.114

#--- Of which the top ten destination addresses were To:
   7145  66.92.70.235
   6610  24.17.25.146
   6196  24.16.155.180
   6076  24.43.12.34
   5597  65.0.39.132
   5097  204.210.138.197
   4937  64.180.86.74
   4625  24.252.125.150
   4236  24.202.11.107
```

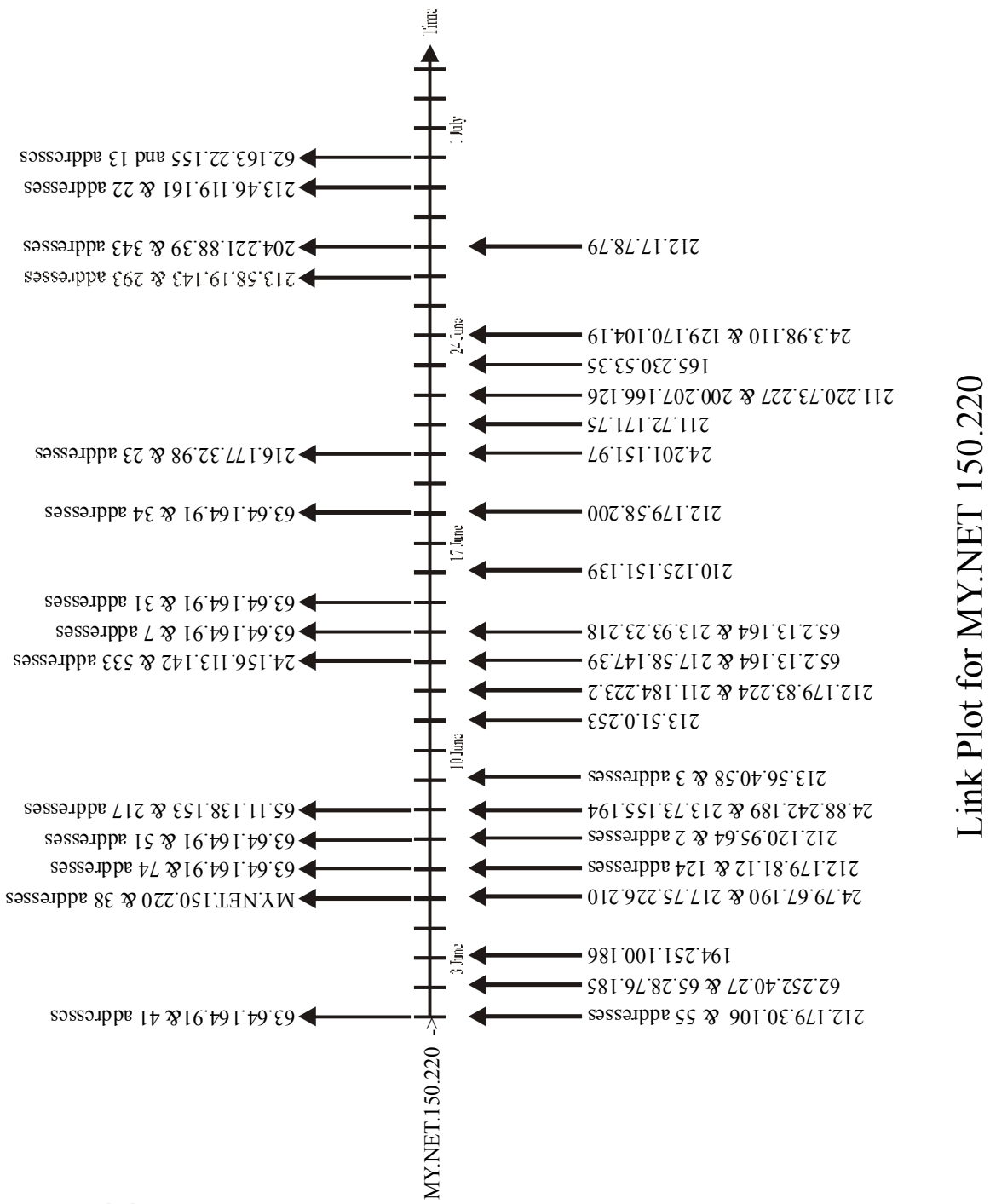
```
   3957  24.68.214.133
#--- The top ten destination ports were:
 115309  27005
   7145  27500
   3179  27243
   1613  2213
   1053  21439
   1045  22952
   1041  24979
    996  21089
    947  14673
    921  21069
```

MY.NET.150.220

The computer MY.NET.150.220, suffered as a target for most of the month. In addition, it appeared as a source in many portscans and alerts. It should be looked at for signs of compromise.

```
#------------------------------------------------------------------------
#---- Monthly Communication Stats for MY.NET.150.220
#------------------------------------------------------------------------

#--- There were 45 src addresses and 815 src ports involved with MY.NET.150.220

#--- Of which the top ten source addresses were From:
 127124  212.179.58.200
  24191  MY.NET.150.220
     35  24.79.67.190
     18  212.179.81.12
     11  65.2.13.164
      8  212.179.30.106
      6  211.220.73.227
      5  212.179.84.89
      4  66.27.72.46
      4  62.252.40.27
#--- The top ten source ports were:
  94489  3697
  32635  3620
  22603  28800
    323  2345
     79  2385
     32  1847
     13  2314
      8  3339
      7  6699
      7  2390

#--- There were 1487 des addresses and 172 des ports involved with MY.NET.150.220

#--- Of which the top ten destination addresses were To:
 127270  MY.NET.150.220
    253  62.248.32.111
    246  24.156.113.142
```

```
   193  137.226.141.184
   185  24.160.141.26
   182  213.113.113.202
   180  65.11.138.153
   176  216.232.117.138
   172  63.57.141.9
   165  62.54.19.179
#--- The top ten destination ports were:
 127173  1234
  20425  28800
    644  8888
    322  6257
    185  1214
    146  7777
    123  6699
     79  2304
     61  1891
     53  1702
```



MY.NET.150.133
MY.NET.150.225
MY.NET.150.204

The computers MY.NET.150.133, MY.NET.150.225, and MY.NET.150.204, suffered as fate qualitatively similar to MY.NET 150.114. They should likewise be investigated for signs of compromise.

```
#------------------------------------------------------------------------
#---- Monthly Communication Stats for MY.NET.150.133
#------------------------------------------------------------------------

#--- There were 257 src addresses and 598 src ports involved with MY.NET.150.133

#--- Of which the top ten source addresses were From:
 124486  MY.NET.150.133
     60  66.72.115.95
     59  212.179.4.50
     57  212.179.27.6
     40  213.10.221.182
     38  212.179.82.238
     36  212.179.81.36
     33  212.179.84.222
     23  212.179.83.69
     21  212.179.127.40
#--- The top ten source ports were:
 122745  28800
    588  3060
    578  3052
    161  1419
     70  2382
     65
     45  2306
     23  2474
     18  3451
     18  1247

#--- There were 1954 des addresses and 523 des ports involved with MY.NET.150.133

#--- Of which the top ten destination addresses were To:
   2573  203.168.199.138
   2536  24.167.51.143
   2389  61.183.120.174
   2208  210.200.167.41
   2162  200.191.17.58
   2137  66.20.77.197
   2084  208.180.106.54
   2015  24.201.39.17
   2009  24.240.221.236
   2004  200.171.233.236
#--- The top ten destination ports were:
 115537  28800
    952  1214
    265  3479
    189  1664
    172  1052
    165  1024
```

```
   150  1087
   145  2296
   145  1071
      142    2148

#-------------------------------------------------------------------
#---- Monthly Communication Stats for MY.NET.150.225
#-------------------------------------------------------------------

#--- There were 134 src addresses and 347 src ports involved with MY.NET.150.225

#--- Of which the top ten source addresses were From:
 128306  MY.NET.150.225
    75  193.226.113.248
    32  212.179.81.73
    29  212.179.34.114
    22  212.179.83.72
    18  212.179.83.109
    17  212.179.82.216
    15  212.179.80.148
    14  212.179.81.114
    12  212.179.82.147
#--- The top ten source ports were:
  85104  28800
  18501  2109
  17712  2102
   4833  3186
   1560  2089
    255  2354
     75  2308
     49  3161
     45  2379
     32  2330

#--- There were 1964 des addresses and 526 des ports involved with MY.NET.150.225

#--- Of which the top ten destination addresses were To:
   1365  24.78.135.13
   1203  217.136.32.88
   1151  24.160.141.26
   1086  213.113.113.202
    975  213.7.4.114
    954  62.211.61.158
    950  151.25.142.50
    884  24.167.51.143
    827  24.226.152.203
    771  61.143.212.186
#--- The top ten destination ports were:
 109350  28800
    586  1214
    458  1026
    302  1127
    294  1350
    292  21024
```

```
   290  4390
   280  1746
   264  1066
      256    1024

#-------------------------------------------------------------------
#---- Monthly Communication Stats for MY.NET.150.204
#-------------------------------------------------------------------

#--- There were 14 src addresses and 16 src ports involved with MY.NET.150.204

#--- Of which the top ten source addresses were From:
  37506  MY.NET.150.204
      3  217.75.226.210
      3  129.170.104.19
      2  213.93.23.218
      2  213.56.40.58
      2  139.134.102.192
      1  61.13.106.35
      1  217.96.196.117
      1  217.81.194.157
      1  217.80.206.28
#--- The top ten source ports were:
  25530  28800
  11967  1403
      8  2328
      3  2225
      3  1969
      2  3516
      2  3135
      2  3084
      2  21
      1  4653

#--- There were 457 des addresses and 120 des ports involved with MY.NET.150.204

#--- Of which the top ten destination addresses were To:
    627  212.156.201.103
    603  210.200.167.41
    575  204.221.88.39
    496  172.146.116.94
    484  64.123.58.51
    475  63.38.64.110
    475  172.160.28.15
    465  149.225.84.128
    460  202.129.238.207
    422  66.37.135.177
#--- The top ten destination ports were:
  34455  28800
    234  21027
    142  4671
    120  1024
     89  1575
     88  1056
```

67 1203
67 1189
66 1490
63 1992

212.179.58.200

On 18 June from 11:00 to 18:00, the computer MY.NET.150.220, undertook a lengthy conversation with 212.179.58.200. This computer is in the "Watchlist 000220 IL-ISDNNET-990517". The name of this Israeli company is "NV-PICTUREVISION". This suggests the possibility that this is benign streaming video, perhaps audio. Nonetheless, the computer disserves scrutiny by virtue of it being on the watchlist.

```
#------ 06/18 11:00 -> 27136 Detects
06/18 11:00    From:
   212.179.58.200       27136
06/18 11:00    To:
   MY.NET.150.220       27136

#------ 06/18 12:00 -> 16010 Detects
06/18 12:00    From:
   212.179.58.200       16010
06/18 12:00    To:
   MY.NET.150.220       16010

#------ 06/18 13:00 -> 11787 Detects
06/18 13:00    From:
   212.179.58.200       11787
06/18 13:00    To:
   MY.NET.150.220       11787

#------ 06/18 14:00 -> 22028 Detects
06/18 14:00    From:
   212.179.58.200       22028
06/18 14:00    To:
   MY.NET.150.220       22028

#------ 06/18 15:00 -> 11473 Detects
06/18 15:00    From:
   212.179.58.200       11473
06/18 15:00    To:
   MY.NET.150.220       11473

#------ 06/18 16:00 -> 11180 Detects
06/18 16:00    From:
   212.179.58.200       11180
06/18 16:00    To:
   MY.NET.150.220       11180

#------ 06/18 17:00 -> 10102 Detects
06/18 17:00    From:
   212.179.58.200       10102
```

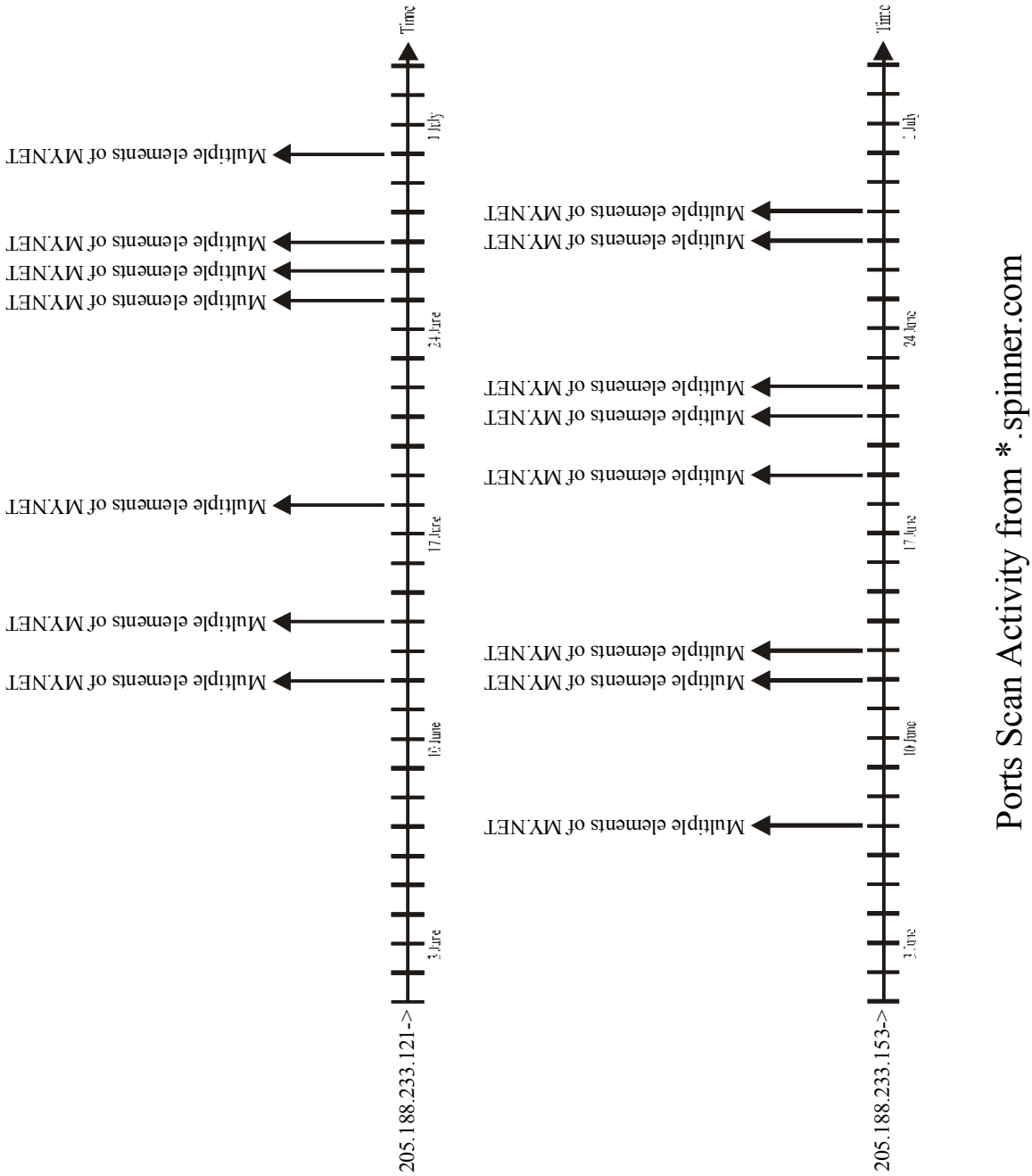
```
06/18 17:00    To:
   MY.NET.150.220       10102

#------ 06/18 18:00 -> 17408 Detects
06/18 18:00    From:
   212.179.58.200       17408
06/18 18:00    To:
   MY.NET.150.220       17408
```

205.188.233.153
205.188.233.121

These two outside computers have been aggressively port scanning MY.NET. Perhaps this subnet should be blocked at the firewall. Port 6970 is associated with the trojan GateCrasher.

```
#-------------------------------------------------------------------------
#---- Monthly Communication Stats for 205.188.233.121
#-------------------------------------------------------------------------

#--- There were 1 src addresses and 709 src ports involved with 205.188.233.121

#--- Of which the top ten source addresses were From:
  36801  205.188.233.121
#--- The top ten source ports were:
   346  23416
   154  8414
   150  28348
   150  22054
   148  10220
   146  21692
   145  13806
   141  25718
   141  15164
   140  22354

#--- There were 33 des addresses and 5 des ports involved with 205.188.233.121

#--- Of which the top ten destination addresses were To:
  3114  MY.NET.106.178
  3080  MY.NET.178.154
  3062  MY.NET.109.62
  3014  MY.NET.110.33
  2461  MY.NET.104.127
  2224  MY.NET.108.13
  2179  MY.NET.108.15
  1932  MY.NET.107.4
  1862  MY.NET.110.169
  1769  MY.NET.15.223
#--- The top ten destination ports were:
 36608  6970
   186  6972
     4  7084
```

```
     2  7080
     1    7082

#-------------------------------------------------------------------------
#---- Monthly Communication Stats for 205.188.233.153
#-------------------------------------------------------------------------

#--- There were 1 src addresses and 778 src ports involved with 205.188.233.153

#--- Of which the top ten source addresses were From:
  39390  205.188.233.153
#--- The top ten source ports were:
    225  26512
    191  15860
    159  8562
    157  27794
    155  17530
    154  27558
    153  19084
    148  16208
    144  16902
    144  16206

#--- There were 31 des addresses and 2 des ports involved with 205.188.233.153

#--- Of which the top ten destination addresses were To:
   3901  MY.NET.108.15
   3635  MY.NET.70.92
   3406  MY.NET.110.33
   3074  MY.NET.178.154
   2844  MY.NET.145.197
   2822  MY.NET.107.4
   2439  MY.NET.110.169
   2324  MY.NET.108.13
   1896  MY.NET.178.222
   1773  MY.NET.145.166
#--- The top ten destination ports were:
  39247  6970
    143  6972
```



211.240.28.66
193.253.243.190

These were very large attempts to communicate with ftp services in MY.NET by 211.240.28.66 and 193.253.243.190. There were 10s of thousands of ftp attempts, a few windows netbios attempts, and they were targeted to over 58,000 IP addresses within MY.NET.

#----------

```
#---- Monthly Communication Stats for 211.240.28.66
#------------------------------------------------------------------------

#--- There were 3 src addresses and 81 src ports involved with 211.240.28.66

#--- Of which the top ten source addresses were From:
  45310  211.240.28.66
      4  169.254.101.152
      2  MY.NET.150.139
#--- The top ten source ports were:
  45233  21
      4  137
      1  2910
      1  2907
      1  2904
      1  2846
      1  2815
      1  2799
      1  2789
      1  2753

#--- There were 16589 des addresses and 2 des ports involved with 211.240.28.66

#--- Of which the top ten destination addresses were To:
      6  MY.NET.105.91
      6  211.240.28.66
      5  MY.NET.100.59
      5  MY.NET.100.165
      5  MY.NET.100.153
      4  MY.NET.99.104
      4  MY.NET.71.15
      4  MY.NET.70.92
      4  MY.NET.70.68
      4  MY.NET.70.27
#--- The top ten destination ports were:
  45312  21
      4    137

#------------------------------------------------------------------------
#---- Monthly Communication Stats for 193.253.243.190
#------------------------------------------------------------------------

#--- There were 1 src addresses and 3944 src ports involved with 193.253.243.190

#--- Of which the top ten source addresses were From:
  43135  193.253.243.190
#--- The top ten source ports were:
     31  3538
     30  3542
     29  3540
     28  3554
     28  3536
     28  3532
     26  3959
```

```
   26  3895
   26  3582
   26  3566

#--- There were 16983 des addresses and 1 des ports involved with 193.253.243.190

#--- Of which the top ten destination addresses were To:
    3  MY.NET.99.97
    3  MY.NET.99.95
    3  MY.NET.99.93
    3  MY.NET.99.91
    3  MY.NET.99.87
    3  MY.NET.99.8
    3  MY.NET.99.77
    3  MY.NET.99.75
    3  MY.NET.99.71
    3  MY.NET.99.65
#--- The top ten destination ports were:
 43135  21
```

The following table enumerates the mal-formed TCP packet flags incident upon MY.NET during the month of June 2001. Most analysts know about SF, null scans, Christmas trees, and reserved bits; but this is an impressive list of exotic flag settings. These results are obtained from the portscans and OOS files.

Alert Count	Mal Formed Flags	Alert Count	Mal Formed Flags	Alert Count	Mal Formed Flags	Alert Count	Mal Formed Flags
44511	**SF****	16	21S*R*AU	11	*1SF****	7	***FRPAU
7726	21S*****	15	**SF**AU	11	*1SF**AU	7	*1SF***U
4396	RESERVEDBITS	15	**SFRPA*	11	*1SFR*AU	7	*1SF**A*
460	********	15	21*FR***	11	2*SFR***	7	*1SFRPA*
94	***FR*A*	15	21S***AU	11	21*F**A*	7	*1SFRPAU
93	**S*R*A*	15	21S**P*U	11	21*FRP**	7	2*SF**A*
79	**SFRP*U	15	21S*R***	11	21S**PA*	7	21**RP**
37	***F****	15	21SF****	10	***FRP**	7	21*FR**U
28	****RP**	15	21SF*PA*	10	*1SFRP*U	7	21S***A*
28	**SFR**U	15	21SF*PAU	10	21**RPAU	7	21S**PAU
25	2*SFR*AU	15	21SFR**U	10	21*F***U	7	21S*RP**
25	21*F****	15	21SFR*AU	10	21*FR*A*	7	21S*RP*U
24	***FR***	15	21SFRPAU	10	21*FRPA*	7	21SF**AU
24	21*F*PA*	14	2*SFRPA*	10	21SF**A*	6	*****P*U
23	**SF***U	14	21S*R**U	9	*1SF*P**	6	***FRP*U
23	**SF*PAU	14	21S*RPA*	9	*1SF*PA*	6	**S*RP**
23	2*SFRP*U	14	21S*RPAU	9	*1SFR*A*	6	*1SFR***

23	21*FR*AU	13	*1SF*P*U	9	2*SFR*A*	6	21**R***
21	**S*RP*U	13	2*SF*PA*	9	2*SFRPAU	6	21S*R*A*
21	**SF*PA*	13	2*SF*PAU	9	21*F**AU	5	**S****U
21	2*SF***U	13	21*F*P**	9	21*F*P*U	5	**S**PA*
20	21*FRPAU	13	21SF***U	9	21*FRP*U	5	**S*RPA*
19	***F*P*U	13	21SFR*A*	9	21SF*P**	5	**S*RPAU
19	**SF**A*	12	*******U	9	21SFR***	4	**S**P**
19	**SF*P**	12	**S***AU	9	21SFRP**	4	**S**P*U
19	**SFR***	12	**SFR*AU	9	21SFRP*U	4	**S*R*AU
19	2*SF****	12	*1SFRP**	9	21SFRPA*	3	****R**U
18	****RP*U	12	2*SF**AU	8	**SFRP**	3	****R*AU
18	***FRPA*	12	2*SF*P*U	8	*1SFR**U	3	****RPAU
18	21**RPA*	12	21*F*PAU	8	2*SF*P**	3	***F***U
17	**SF*P*U	12	21S**P**	8	2*SFR**U	3	***FR**U
17	2*SFRP**	11	***F*P**	8	21**R**U	3	**S**PAU
17	21**R*AU	11	**S*R***	8	21**RP*U	2	***FR*AU
16	*1SF*PAU	11	**SFR*A*	8	21S****U	2	**S*R**U
16	21**R*A*	11	**SFRPAU	8	21SF*P*U		

In addition to traffic with mal-formed TCP flags directed at the network, there was significant amount of traffic found with exotic TCP options set. Some particularly aggregious example are shown below:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/22-18:30:59.182025 192.168.1.1:0 -> 216.235.163.151:0
TCP TTL:93 TOS:0x94 ID:1062
IP Options => ^@EOL EOL EOL EOL
21S**P** Seq: 0xAFE6500   Ack: 0xEFF8E700   Win: 0x0
TCP Options => EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL
EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EO
L EOL EOL EOL EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/05-02:19:29.367812 24.29.186.167:3872 -> MY.NET.217.62:61327
TCP TTL:18 TOS:0x0 ID:17547  DF
*1SFRP** Seq: 0xF18FF18F   Ack: 0xF28FF48F   Win: 0xFC8F
TCP Options => SackOK SackOK WS: 15 EOL CCECHO: 583322423 CCECHO: 583322423
CCECHO: 583322423 CCECHO: 583322423 CCECHO: 583322423
 CCECHO: 583322423 CCECHO: 583322423 CCECHO: 583322423 CCECHO: 583322423
CCECHO: 583322423 CCECHO: 583322423 CCECHO: 583322423 CC
ECHO: 583322423 CCECHO: 583322423 CCECHO: 583322423 CCECHO: 583322423
CCECHO: 583322423 CCECHO: 583322423 CCECHO: 583322423 CCECH
O: 583322423 CCECHO: 583322423 CCECHO: 583322423 CCECHO: 583322423 CCECHO:
583322423 CCECHO: 583322423 CCECHO: 583322423 CCECHO:
583322423 CCECHO: 583322423 CCECHO: 583322423 CCECHO: 583322423 CCECHO:
583322423 CCECHO: 583322423 CCECHO: 583322423 CCECHO: 583
322423 CCECHO: 583322423 CCECHO: 583322423

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/22-07:31:03.671206 211.220.73.227:2096 -> MY.NET.150.220:1234
```

```
TCP TTL:112 TOS:0x0 ID:1931  DF
*1SFR*AU Seq: 0x920001   Ack: 0xAD2022C3   Win: 0x5010
TCP Options => Opt 45 Opt 45 Opt 45 Opt 45 Opt 45 Opt 45 Opt 45 Opt 45 Opt 45 Opt
45 Opt 45 Opt 45 Opt 45 Opt 45 Opt 45 Op
t 45 Opt 45 Opt 45 Opt 45 Opt 45 Opt 45 Opt 45 Opt 45 Opt 45 Opt 45 Opt 45 Opt 45
Opt 45 Opt 45 Opt 45 Opt 45 Opt 45 Opt 4
     5    Opt 45 Opt 45 Opt 45 Opt 45 Opt 45

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/12-13:52:49.696675 62.180.222.141:32798 -> MY.NET.150.133:9824
TCP TTL:112 TOS:0x0 ID:60503  DF
21SF***U Seq: 0xFAE0E80D   Ack: 0x11D4DEC1   Win: 0xA522
TCP Options => Opt 255 (11): A6B9 8651 8C09 A981 82B3 Opt 179 Opt 179 Opt 179 Opt 179
Opt 179 Opt 179 Opt 179 Opt 179 Opt 179 Opt
 179 Opt 179 Opt 179 Opt 179 Opt 179 Opt 179 Opt 179 Opt 179 Opt 179 Opt 179 Opt
179 Opt 179 Opt 179 Opt 179 Opt 179 Opt
179 Opt 179 Opt 179 Opt 179 Opt 179 Opt 179 Opt 179 Opt 179 Opt 179 Opt 179 Opt
179 Opt 179 Opt 179 Opt 179 Opt 179 Opt 179
```

This traffic is represetative of that used by hostiles to exercise TCP stacks in order to gather information, e.g OS finger printing, or damage or to confuse services and perhaps disrupt rsevices running on potential targets.

Defensive Recommendations

If the university does not have one, it needs an "appropriate computer use" policy that is unambiguous and rigorously enforced. A few high profile disciplinary actions on the part of the university might well help in discouraging hostile activity that might emanate from personnel and students within the university.

In my institution, we use the SARA scanning tools (http://www-arc.com/sara/index.shtml) to make regular scans of the entire network looking for vulnerable services and misconfigured computational resources. Tools like this may enable institutions like universities to proactively deal with security problems before they develop into security incidents. This course of action is high-maintenance, intrusive on sysadmins limited time resources, and requires significant additional personnel resources. It is, however, effective in discovering and addressing problems before they become crises.

I recommend that certain ports be blocked at the university firewall. Certainly the RPC ports (111) and perhaps the lpd port (515).

There are several inside the university IP subnets associated with a great deal of hostile activity as seen by the IDS. There are also a large number of alerts that have derived from that traffic These subnets are the sources of significant internally sourced portscan and hostile activity. Perhaps some closer supervision of these nets is warranted.

MY.NET.150.*
MY.NET.160.*
MY.NET.98.*

MY.NET.60.*
MY.NET.97.*

The following computers, in particular, have been involved in significant hostile traffic as detected by the IDS. This activity suggests that many of these computers may be root compromised. I recommend that all these computer be looked at for signs of compromise or trojan infection.

MY.NET.104.111	MY.NET.150.133	MY.NET.253.24
MY.NET.106.178	MY.NET.150.133	MY.NET.6.34
MY.NET.107.4	MY.NET.150.204	MY.NET.6.47
MY.NET.108.13	MY.NET.150.220	MY.NET.60.16
MY.NET.108.15	MY.NET.150.225	MY.NET.69.209
MY.NET.109.62	MY.NET.160.114	MY.NET.70.242
MY.NET.110.169	MY.NET.160.169	MY.NET.70.38
MY.NET.110.33	MY.NET.163.54	MY.NET.70.92
MY.NET.111.139	MY.NET.178.154	MY.NET.70.97
MY.NET.130.122	MY.NET.178.222	MY.NET.97.175
MY.NET.144.59	MY.NET.182.120	MY.NET.97.210
MY.NET.145.166	MY.NET.218.198	MY.NET.99.51
MY.NET.145.197	MY.NET.218.82	
MY.NET.146.95	MY.NET.253.24	

These external IP addresses have undertaken enormous, apparently hostile, traffic with a few internal IP addresses.. Perhaps they should be blocked at the firewall.

156.226.0.0
193.253.243.190
205.188.233.121
205.188.233.153
205.188.233.185
205.188.244.121
205.188.244.249
205.188.246.121
211.235.241.145
211.240.28.66
212.179.0.0
212.179.58.200
212.179.79.2
94.87.6.255

Analysis Procedure

"*If the only tool you have is a hammer, you tend to see every problem as a nail.*"

. . . . Abraham Maslow

I am good at shell scripting, awk, sed, vi, and Microsoft Excel. I chose those tools to "Analyze This". Apart from the aggravating process of transferring data back and forth between my linux IDS analysis box where I run snort, and my office PC on which I run Excel, there is a fundamental limitation to Excel in that it that permits only 65,536 rows of data. I can easily generate the comma separated files for immediate input of tcpdump files into Excel spreadsheets. But Excel proved insufficient to the task of looking at the large blocks of data associated with "analyze this". In the end, I only used Excel for preparing columns of data and graphs for this report.

It may be possible to use Excel on my much smaller network (508 IP addresses soon double). However, the ftp file transfer is a considerable headache, and I can easily imagine running into future problems associated with the Excel row constraints.

Over the last three months and throughout the GIAC "Intrusion Detection In Depth" course especially in the snort portion, I learned of what seems to be a sensible avenue of approach to this problem in using SQL databases. These exist for Linux, and perhaps I need to put learning about databases on my B+ list of things to do. (It is something I have actively avoided for years….alas.)

I built shell scripting tools to analyze my own tcpdump/snort data, and used them for this exercise. The tools went through two or three revisions, as I first wrote them for analysis of data on my own network, then rewrote them for the "Analyze This" project, and found out that the improvements were useful for work back on my own network. (This is a good thing.)

I wrote scripts to:

- Gather all the portscans, alerts, and oos files into three large single files (accumulated_portscans, etc).
- For entire data set (all 3 accumulated_* files), gather all the src and dst ports, and the src and dst IP addresses; sort and count them.
- Gather each alert message; sort and count them.
- For each alert message, Gather all the src and dst ports and IP addresses associated with that port; sort and count them.
- I built a complicated script that follows one IP address through out the one month period. The purpose of this script was to help in building the time line link maps. It extracts all the detects from alerts, portscans, and oos files for one IP as either source or dst. While the IP address is a source, it gathers, sorts, and counts the dst IP addresses and ports. While the IP address is the destination, it gathers, sorts, and counts the src IP addresses and the dst ports. In addition it does this three times and creates three separate reports. The first report provides monthly grand totals for all detects associated with the "IP address of interest"; the second provides day by day sub totals of the same information; and the third file provides hour by hour sub totals. These files are rich in detail that permits one to step back

and get a good overview of the activity of one address. I was astonished at how much clearer my understanding of the data became after doing this.

The results of this script have suggested to me the outline of an interesting project that I will be working on in the very near future. I think that I can expand the utility of this tool to obtain iterative information. By which I mean, select an IP address -> gather its monthly, daily, and hourly statistics -> this will suggest other interesting IP address for further study -> return to step 1. Then figure out a way to automatically graph the multi-IP address information (which I failed to do for the "Analyze This" exercise). As much as I love and use Excel, it's graphing capabilities are terrible (really, really terrible). But I think I can create useful output files that can be imported into Matlab (a high-end number crunching platform that I use). Matlab has grown-up graphing capabilities that I think I can use it to present this information in a clarifying manner. More important, I think I can make this a nearly automated tool, that can be used on the fly when an interesting IP address pops up in the daily snort files.

In addition I have built some other tools, whose utility have grown as they were built for my own use, modified for "Analyze This", and/or rebuilt for my use>

- processIP: This script is basically a "grep IPaddress detect.logs" command. There are multiple lines per detect issues, and there are date format translation issues requiring sed, awk etc. So it is a bit more complicated than the grep command suggests.
- processPortscan: Gathers all portscans for a target portscan.log; then it sorts and segregates by src IP, sorts by time, performs nslookup. This script identifies and organizes, and isolates all portscan activity from a portscan.log. It makes the portscan.log readable and useful.
- processAlert: This script greps for the "[**]" element of an alert file. It sorts and counts alerts. It is the first script I run every morning to see what happened yesterday.
- cull: This script is always run following processAlert. It gathers all the alerts associated with an alert header.

For example:

What alerts happened on July 20?

```
[root@liar ids]# ./processAlert July20
   332 [**] ICMP Destination Unreachable [**]
   318 [**] ICMP Time Exceeded (Undefined Code!) [**]
    27 [**] SCAN-SYN FIN [**]
    14 [**] RPC portmap request mountd [**]
    14 [**] RPC Info Query [**]
     6 [**] MISC traceroute [**]
     . . . And another 30 lines . . .
```

There were 27 SYN-FIN detects that occurred on July 20. Who is responsible for this?

```
[root@liar ids]# ./cull "SCAN-SYN FIN" July20
[**] SCAN-SYN FIN [**]
07/20-19:35:23.104940 211.251.138.129:600 -> mynet.org.4.33:600
TCP TTL:15 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x5E77F2E  Ack: 0xB674CE  Win: 0x404  TcpLen: 20


[**] SCAN-SYN FIN [**]
07/20-19:35:24.884754 211.251.138.129:600 -> mynet.org.4.122:600
TCP TTL:15 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x61EE5A32  Ack: 0x2AC80AC7  Win: 0x404  TcpLen: 20

    . . . And there are 25 more just like these  . . .
```


Appendix

It is "padding" to present all the scripts; I think it is more important to describe the functionality of the scripts as done above. I will show only one so that the reader can see the way that all of these scripts work. The script I have selected is processPortscan. I use it several times a day, it is for all practical purposes finished. Note: I have the habit of writing code with long lines on my 20 inch monitor. It does not copy and paste well into the 8.5x11 inch word processing format. I tried to edit this with "\" in the right places. I hope they are all there.

```
[root@liar ids]# cat processPortscan
#!/bin/sh

#--- Sanity Checking
if [ $# -ne 1 ]
then
  echo "
     This script organizes portscan.log into a more useful document.

     Grammar:
       ./processPortscan July04
  "
  exit
fi
```

```
if [ ! -d /home/ids/$1 ]
then
  echo "
     Time to work on out cluefulness skills....

     The directory /home/ids/$1,
     actually has to exist, for this to work.

     ./processPortscan July04
  "
  exit
fi

source=$1
dir="/home/ids"

#----- Main -------------------------------------------------------
cp /dev/null  acc_portscan.log
cat $dir/$source/portscan.log > $dir/acc_portscan.log 2> /dev/null

#--- For now we do not deal with scans from inside network
cat $dir/acc_portscan.log  | awk '{if ($4 !~ /mynet.org.4./) {print \ $0}}' > $dir/foo_not_from_4
cat $dir/foo_not_from_4 | awk '{if ($4 !~ /mynet.org.5./) {print $0}}' \ > $dir/foo_not_from_4_or_5

#--- Make sure there are no double counts
sort $dir/foo_not_from_4_or_5 | uniq  > $dir/portscan.log_sorted

if [ -f $dir/acc_portscan.log ] ; then  rm  $dir/acc_portscan.log; fi
if [ -f $dir/foo_not_from_4 ] ; then  rm $dir/foo_not_from_4 ; fi
if [ -f $dir/foo_not_from_4_or_5 ] ; then  rm $dir/foo_not_from_4_or_5 \ ; fi

#--- There are repeated scans by the same IP address, gather them together

cat $dir/portscan.log_sorted | gawk '{print $4}' | gawk -F: '{print $1}' | sort | uniq >
$dir/temp_hostile

cp /dev/null $dir/portscan.log_sorted_colated

for hostile in `cat $dir/temp_hostile`
do
  echo "#---- Portscan from:   $hostile --------------------------" >> $dir/portscan.log_sorted_colated

  nslookup $hostile > temp_nslookup 2> /dev/null

  egrep -v "^Server:" $dir/temp_nslookup | egrep -v "^Address:" | \   sed 's/^/    /' >>
$dir/portscan.log_sorted_colated
  echo "" >> $dir/portscan.log_sorted_colated

  grep $hostile $dir/portscan.log_sorted | grep  "^Jan " >> \ $dir/portscan.log_sorted_colated
  grep $hostile $dir/portscan.log_sorted | grep  "^Feb " >> \ $dir/portscan.log_sorted_colated
  grep $hostile $dir/portscan.log_sorted | grep  "^Mar " >> \ $dir/portscan.log_sorted_colated
  grep $hostile $dir/portscan.log_sorted | grep  "^Apr " >> \ $dir/portscan.log_sorted_colated
  grep $hostile $dir/portscan.log_sorted | grep  "^May " >> \ $dir/portscan.log_sorted_colated
  grep $hostile $dir/portscan.log_sorted | grep  "^Jun " >> \ $dir/portscan.log_sorted_colated
```

```
  grep $hostile $dir/portscan.log_sorted | grep  "^Jul " >> \ $dir/portscan.log_sorted_colated
  grep $hostile $dir/portscan.log_sorted | grep  "^Aug " >> \ $dir/portscan.log_sorted_colated
  grep $hostile $dir/portscan.log_sorted | grep  "^Sep " >> \ $dir/portscan.log_sorted_colated
  grep $hostile $dir/portscan.log_sorted | grep  "^Oct " >> \ $dir/portscan.log_sorted_colated
  grep $hostile $dir/portscan.log_sorted | grep  "^Nov " >> \ $dir/portscan.log_sorted_colated
  grep $hostile $dir/portscan.log_sorted | grep  "^Dec " >> \ $dir/portscan.log_sorted_colated
  echo "" >> $dir/portscan.log_sorted_colated
  echo "" >> $dir/portscan.log_sorted_colated
done

if [ -f $dir/temp_nslookup ] ; then  rm $dir/temp_nslookup ; fi
if [ -f $dir/temp_nslookup ] ; then  rm $dir/portscan.log_sorted ; fi
if [ -f $dir/portscan.log_sorted ] ; then  rm $dir/portscan.log_sorted \ ; fi
if [ -f $dir/temp_hostile ] ; then  rm $dir/temp_hostile ; fi

cat $dir/portscan.log_sorted_colated  | more
```

Bibliography

http://ouah.bsdjeunz.org/George_Bakos.html
http://www-arc.com/sara/index.shtml
http://www.datafellows.com/v-descs/adore.shtml
http://www.kulua.org/Archives/kulua-l/200008/msg00159.html
http://www.research.umbc.edu/~andy
http://www.sans.org/y2k/040401.htm
http://www.sans.org/y2k/051900.htm
http://www.sans.org/y2k/052000.htm
http://www.sans.org/y2k/063000.htm
http://www.sans.org/y2k/082200.htm
http://www.sans.org/y2k/120600-1200.htm
http://www.snort.org
http://www.whitehats.com/info/IDS442