Intrusion Detection Practical Assignment
For SANS Security DC 2000

By Alan Powell

Introduction

This document is divided into three parts, per the requirements of the Practical Assignment for Intrusion Detection, DC 2000:

The first section covers the Network Detects Assignment. I present five Network Detects with an analysis of each detect by answering the following questions about each trace:

1. Source of trace

2. Detect was generated by:

3. Probability the source address was spoofed

4. Description of attack:

5. Attack mechanism:

6. Correlations:

7. Evidence of active targeting:

8. Severity:

9. Defensive recommendation:

The first two detects are to an attack listed in the SANS Ten Most Critical Internet Security Threats page, http://www.sans.org/topten.htm, the other three detects are representative of some of the attacks/probes we see on our network. For each trace, I also provide a multiple choice test question, based on the trace and my analysis.

Assignment One:

The following detects are from a medium size .edu network. We employ a custom built firewall, which sits between out network and our Service Provider, and some host based intrusion detection tools like BlackIce Defender (http://www.networkice.com) on the PC side and tcp-wrappers (ftp://ftp.porcupine.org/pub/security/index.html) on the unix side. Our traditional approach to Intrusion Detection has been what I would call reactive rather than proactive. The general consensus here has been that there are simply too many probes and attacks being attempted to look into every one. Instead, we have concentrated on trying to create an environment which makes attacks and reconnaissance difficult. This has included things like:

1) Tuning our firewall to prevent common exploits and recognizance and also to drop traffic, at least to critical subnets, that is unnecessary. We may, for example, block all external traffic to our mail server except for traffic to the SMTP port.
2) Re-structuring our network so that it is more secure in the sense of allowing sniffing
3) Increasing host level security. We have started to try and eliminate services that are not being used from being installed or turned on, for example. The more services and daemons running, the more things attackers have to attack and the more we have to try to keep patched and up to date. If something is not being used, eliminate it!
4) Educate our users better regarding security

While I agree with this general philosophy of preventing exploits before they happen, I am also trying to move us towards doing more intrusion detection. While I agree that it's impossible to worry about every probe and potential exploit, I feel it is important to get a feel for what types of (malicious) activities and surveillance is being carried on out on our network. We have being using host based Intrusion Detection mechanisms on our core servers like Mail, DNS, etc for a long time and this has been very useful in alerting us to malevolent behavior, but I would like to see us carry that forward to the network level so we can we get a feel for what is happening on a larger scale.

The names of the hosts in the following detects are obfuscated. I changed the names of hosts in the .edu to some.edu and the name of the outside hosts so that only the root domain is correct, etc. If part of the host name was (potentially) significant in terms of offering information to the analyst, then I try to obfuscate the name is such a way so as to elicit the information. If one the hosts is a DNS server, for example, I may use something like dns1.some.com. I also changed IP numbers to be in the 192.168.x.x range.

I would also like to point out that I use a lot of host based detects as opposed to firewall or tcpdump logs. Since we primarily use host based Intrusion Detection mechanisms at this point, the detects are representative of what I have to work with when investigating exploits, scans and so on. Also, I only assumed responsibility for security matters shortly before attending the SANS DC Conference. Studying for the tests and working on this practical has given me a much broader feel for what took look for Intrusion Detection wise and what tools are available to help in detection. Before the Conference, as you will see, I largely used tcpdump and host based logs as detection tools and really only ran tcpdump in after the break-in situations.

The Detects:

1) Sadmin Buffer Overflow

```
13:43:00.961944 outsider.com.731 > sun1.some.edu.sunrpc:  udp 56
13:43:00.964801 sun1.some.edu.sunrpc > outsider.com.731:  udp 28 (DF)
13:43:00.965311 outsider.com.732 > sun1.some.edu.32922:  udp 1412
13:43:01.427789 sun1.some.edu.32922 > outsider.com.732:  udp 76 (DF)
```

1. Source of trace

The trace is from the .edu network I am basing my detects on.

2. Detect was generated by:

tcpdump running on a host on the same subnet, after the original computer had been broken into.

3. Probability the source address was spoofed

Unlikely. The attacker broke into this computer via the buffer overflow, indicating connections back to the source address. Also, this was not a DOS attack, so it's unlikely the source address was spoofed.

4. Description of attack:

CVE: CVE-1999-0977

I found information on this attack at: http://www2.merton.ox.ac.uk/~security/bugtraq-199912/0145.html

This is one of the first attacks/compromises that I was involved in investigating and I have limited detects for it and very little correlation with other logging sources. It is one of the SANS Top Ten exploits, however, and is typical of the exploits we see in our environment. Our core servers are well maintained and patched and we have had few confirmed break-ins of these servers. Individual departments running their own, standalone workstations, however, are typically not as well maintained and we have seen several break-ins on them, especially sadmin and rpc exploits.

We were alerted by another .edu that several of their Solaris computers had been broken into via the sadmin buffer overflow and that they had reason to be that their compromised machines had been used a launch pads for exploits of our network. We immediately cut off network access to the affected machines and I did forensics on one of them. The attacker had largely removed all logs, which was a sign that the machine had been broken into itself, but I did find the following files:

/usr/lib/libx - dated Jun 14 05:46, empty directory, permission set at 777

/etc/m - dated Jun 14 05:46, contains a copy of /etc/inet/inetd.conf (this file does not contain an ingreslock line)

/etc/core - dated Jun 13 18:10, ran strings on the file and saw references to ingreslock

/tmp/.x - dated Jun 13 14:36, contains " ingreslock stream tcp nowait root /bin/sh sh -i"

Once I realized the machine had, indeed, been broken into, we saved as much evidence as we could and then re-installed the OS, applied all patches, commented everything out of inetd.conf, forced the owner to only install and only use ssh for remote logins and started logging everything we could via tcp-wrappers. I also started running tcpdump with a simple ruleset to look for connections to this machine and I began to see many connections similar to the one above. I searched the Internet for a program which could be used to exploit the sadmin buffer overflow and when I ran it in a controlled environment, I discovered the above signature was typical of an exploit attempt on a machine running sadmin but which had been patched.

5. Attack mechanism:

Sadmind is the daemon used by Solstice AdminSuite applications to perform distributed system administration operations such as adding users. The sadmind daemon is started automatically by the inetd daemon whenever a request to invoke an operation is received:

```
100232/10 tli rpc/udp wait root /usr/sbin/sadmind sadmind
```

Certain versions of Solaris ship with a version of sadmind which are vulnerable to a remotely exploitable buffer overflow attack. Under vulnerable versions of sadmind (2.6 and 7.0 have been tested), if a long buffer is passed to a NETMGT_PROC_SERVICE request (called via clnt_call()), it is possible to overwrite the stack pointer and execute arbitrary code. The actual buffer in questions appears to hold the client's domain name. The overflow in sadmind takes place in the amsl_verify() function. Because sadmind runs as root any code launched as a result will run as with root privileges, therefore resulting in a root compromise.

6. Correlations:

Since this was one of the first exploits I was involved in working on, and happened a couple of months ago, I was not able to get much correlating evidence. The fact that I found the existence of the above files and that the log files had all been erased did correlate to the alert we got from the outside .edu, but the firewall logs were gone by the time I investigated the exploit (we had a log storage capacity problem until we recently upgraded our firewall machine, so logs were only kept for a short period of time), so I have minimal evidence of the actual attack itself.

7. Evidence of active targeting:

Very strong. Since the machine had actually been compromised, a attacker had obviously identified the machine as being vulnerable and target it for attack.

8. Severity: Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures)

Criticality
2 (The compromised host is a Sun workstation in an science department. It is primarily a stand alone host and only a couple of people use it.)
Lethality
4 (The attack succeeded and the attacker installed trojan programs and was possibly sniffing the network for passwords. The attacker could potentially have broken into other systems over time. I might even rate this a 5 except that our core servers only allow ssh connections and these systems were on a subnet with low privilege users; no campus wide administrators passwords could have been captured.)
System Countermeasures
3 (The compromised host was running a older version of Solaris and was missing some patches)
Network Countermeasures
2 (Being an education institution, it's difficult politically to have an aggressive firewall. We are making progress, but we have a fairly permissive firewall at this point.)

Severity = (2 + 4) - (3 + 2) = 2

The calculation shows a severity of 2 but I might even make it a 3. The exploit was a targeted one and compromised a non core system. This potentially allows the attacker to gather passwords, to launch attacks against other sites, which is damaging to our reputation, and gives them a somewhat better chance at getting foothold into one of our core systems.

9. Defensive recommendation:

Block connections to the Portmapper and the range of ports used by Sadmin and RPCs via filtering rulesets on your router(s) or Firewall(s).

Also, unless you require sadmin (if your using the Solstice AdminSuite you do) comment sadmind out from your /etc/inetd.conf entry. By default, the line in /etc/inetd.conf that starts sadmind appears as follows:

```
100232/10 tli rpc/udp wait root /usr/sbin/sadmind sadmind
```

If sadmin is required, running sadmin with the command 'sadmind -S2' and setting the stack to noexec_user_stack =1" via /etc/system is advisable.

10. Multiple choice question:

Question: A recent attack involves a buffer overflow in the Sadmin daemon. This daemon:

A. Runs on linux systems and is a component of the NFS system.
B. Runs on Windows PCs and is used for remote administration.
C. Runs on Solaris computers and is used to perform system administration functions like adding users.
D. Runs on many platforms and is a ftp like service.

Answer is C

2) DNS – Probing of machine in an attempt to compromise BIND?

<u>Trace A – tcpdump dump snippets (with snaplen of 1500)</u>

2000-08-02 tcpdump fragment:

```
21:56:56.236634 dialup.isp.com.1032 > dns-serv1.some.edu.domain:  5665 inv_q+ [b2&3=0x980] A? . (27)
21:56:56.238651 dns-serv1.some.edu.domain > dialup.isp.com.1032:  5665 inv_q Refused [0q] 1/0/0 (27)
21:56:56.238718 dialup.isp.com.1032 > dns-serv1.some.edu.domain:  64104+ [b2&3=0x180] TXT CHAOS)?
version.bind. (30)
21:56:56.240846 dns-serv1.some.edu.domain > dialup.isp.com.1032:  64104* 1/0/0 CHAOS) TXT 8.2.2-P5 (63)
```

2000-08-06 tcpdump fragment:

```
14:40:12.457210 cable.modem.ca.2386 > dns-serv1.some.edu.echo: . ack 1 win 32120 (DF)
14:40:12.677082 cable.modem.ca.2598 > dns-serv1.some.edu.32774: S 2422570057:2422570057(0) win 32120
<mss 1460,sackOK,timestamp 183800356 0,nop,wscale 0> (DF)
14:40:12.677159 cable.modem.ca.2599 > dns-serv1.some.edu.32780: S 2427875175:2427875175(0) win 32120
<mss 1460,sackOK,timestamp 183800356 0,nop,wscale 0> (DF)
14:40:12.721900 dns-serv1.some.edu.shell > cable.modem.ca.2464: F 1:1(0) ack 2 win 64240
14:40:12.721943 cable.modem.ca.2464 > dns-serv1.some.edu.shell: . ack 2 win 32120 (DF)
14:40:12.978879 dns-serv1.some.edu.login > cable.modem.ca.1977: P 1:31(30) ack 2 win 64240
14:40:12.978933 cable.modem.ca.1977 > dns-serv1.some.edu.login: R 2437521089:2437521089(0) win 0
14:40:12.979012 dns-serv1.some.edu.login > cable.modem.ca.1977: F 31:31(0) ack 2 win 64240
14:40:12.979050 cable.modem.ca.1977 > dns-serv1.some.edu.login: R 2437521089:2437521089(0) win 0
14:40:12.997076 cable.modem.ca.2600 > dns-serv1.some.edu.32774: S 2425491553:2425491553(0) win 32120
<mss 1460,sackOK,timestamp 183800388 0,nop,wscale 0> (DF)
14:40:12.997152 cable.modem.ca.2601 > dns-serv1.some.edu.32780: S 2427372157:2427372157(0) win 32120
<mss 1460,sackOK,timestamp 183800388 0,nop,wscale 0> (DF)
14:40:13.317068 cable.modem.ca.2602 > dns-serv1.some.edu.32774: S 2436609433:2436609433(0) win 32120
<mss 1460,sackOK,timestamp 183800420 0,nop,wscale 0> (DF)
14:40:13.317144 cable.modem.ca.2603 > dns-serv1.some.edu.32780: S 2429649269:2429649269(0) win 32120
<mss 1460,sackOK,timestamp 183800420 0,nop,wscale 0> (DF)
14:40:13.657398 cable.modem.ca.34998 > dns-serv1.some.edu.echo: S [ECN-Echo] 1365318603:1365318603(0)
win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
14:40:13.657427 cable.modem.ca.34999 > dns-serv1.some.edu.echo: . win 4096 <wscale 10,nop,mss
265,timestamp 1061109567 0,eol>
14:40:13.657480 cable.modem.ca.35000 > dns-serv1.some.edu.echo: SFP 1365318603:1365318603(0) win 4096
urg 0 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
14:40:13.657533 cable.modem.ca.35001 > dns-serv1.some.edu.echo: . ack 0 win 4096 <wscale 10,nop,mss
265,timestamp 1061109567 0,eol>
14:40:13.657585 cable.modem.ca.35002 > dns-serv1.some.edu.tcpmux: S 1365318603:1365318603(0) win 4096
<wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
```

```
14:40:13.657663 cable.modem.ca.35003 > dns-serv1.some.edu.tcpmux: . ack 0 win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
14:40:13.657687 cable.modem.ca.35004 > dns-serv1.some.edu.tcpmux: FP 1365318603:1365318603(0) win 4096 urg 0 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
14:40:13.657773 cable.modem.ca.34991 > dns-serv1.some.edu.1:  udp 300
```

Trace B - From /var/adm/messages on dns-serv1.some.edu

```
Jul 22 00:07:49 dns-serv1 named[5950]: unapproved update from
[192.168.x.x].14180 for dyn.some.edu

Jul 29 02:48:10 dns-serv1 named[5692]: unapproved AXFR from [144.x.x.x].45500 for "some.edu" (acl)
Jul 29 04:12:11 dns-serv1 named[5692]: unapproved AXFR from [144.x.x.x].46041 for "168.192.in-addr.arpa" (acl)
Jul 29 14:28:47 dns-serv1 named[5692]: dropping source port zero packet from [216.x.x.x].0
Jul 29 14:28:50 dns-serv1 last message repeated 2 times
Jul 29 14:28:56 dns-serv1 named[5692]: dropping source port zero packet from [216.x.x.x].0
Jul 29 14:29:08 dns-serv1 last message repeated 5 times
Jul 29 14:29:12 dns-serv1 named[5692]: unapproved AXFR from [144.x.x.x].47876 for "168.192.in-addr.arpa" (acl)
Jul 29 14:29:14 dns-serv1 named[5692]: dropping source port zero packet from [216.x.x.x].0
Jul 29 14:29:17 dns-serv1 last message repeated 2 times
Jul 29 14:35:37 dns-serv1 named[5692]: dropping source port zero packet from [216.x.x.x].0
Jul 29 14:35:48 dns-serv1 last message repeated 4 times
Jul 29 14:35:49 dns-serv1 named[5692]: dropping source port zero packet from [216.x.x.x].0
Jul 29 14:35:55 dns-serv1 named[5692]: dropping source port zero packet from [216.x.x.x].0
Jul 29 14:35:58 dns-serv1 last message repeated 2 times
 …
Jul 30 21:45:39 dns-serv1 named[5692]: unapproved AXFR from [persistent.com].4330 for "subdom1.some.edu" (acl)
Jul 30 21:45:39 dns-serv1 named[5692]: unapproved AXFR from [persistent.com].4335 for " subdom2.some.edu " (acl)
Jul 30 21:45:40 dns-serv1 named[5692]: unapproved AXFR from [persistent.com].4337 for " subdom3.some.edu " (acl)
Jul 30 21:45:40 dns-serv1 named[5692]: unapproved AXFR from [persistent.com].4339 for " subdom4.some.edu " (acl)
Jul 30 21:45:40 dns-serv1 named[5692]: unapproved AXFR from [persistent.com].4341 for " subdom5.some.edu " (acl)
…
Aug 06 12:39:08 dns-serv1 sshd[21440]: error: setsid: Not owner
Aug 06 13:36:09 dns-serv1 named[6280]: dropping source port zero packet from [209.x.x.x].0
Aug 06 13:36:14 dns-serv1 last message repeated 2 times
Aug 06 13:53:40 dns-serv1 named[6280]: reloading nameserver
Aug 06 13:53:40 dns-serv1 named[6280]: Ready to answer queries.
Aug 06 14:39:16 dns-serv1 unix: bpf: lo0 attached
Aug 06 14:39:16 dns-serv1 unix: bpf: en0 attached
Aug 06 14:40:12 dns-serv1 inetd[3626]: accept (for echo): Software caused connection abort
Aug 06 14:40:12 dns-serv1 inetd[3626]: accept (for discard): Software caused connection abort
Aug 06 14:40:12 dns-serv1 inetd[3626]: accept (for chargen): Software caused connection abort
Aug 06 18:40:12 dns-serv1 inetd[5432]: accept: Software caused connection abort
Aug 06 14:40:12 dns-serv1 snmpd[4902]: FATAL: join_tcp_client: Software caused connection abort
Aug 06 14:40:12 dns-serv1 snmpd[4902]: NOTICE: logging to /usr/tmp/snmpd.log is terminating
Aug 06 14:40:12 dns-serv1 identd[22714]: accept() failed: Software caused connection abort
Aug 06 14:40:12 dns-serv1 inetd[3626]: /usr/local/sbin/identd: exit status 0x100
Aug 06 18:40:12 dns-serv1 inetd[5432]: accept: Software caused connection abort
Aug 06 18:40:13 dns-serv1 inetd[5432]: accept: Software caused connection abort
Aug 06 14:40:13 dns-serv1 inetd[3626]: accept (for dtspc): Software caused connection abort
Aug 06 14:40:13 dns-serv1 inetd[3626]: accept (for time): Software caused connection abort
Aug 06 14:40:13 dns-serv1 inetd[3626]: accept (for daytime): Software caused connection abort
Aug 06 14:40:13 dns-serv1 inetd[3626]: accept (for login): Software caused connection abort
```

1. Source of trace

Both traces are from the .edu network I am basing my detects on. Trace A is comprised of two tcpdump fragments which represent some unusual activity amid the normal DNS traffic. As I indicated in my introduction, we are not currently actively using a network IDS system; We log unusual activity via our firewall log for after the fact investigations only. I would like to see us move to more active detection, however, so after coming back from SANS DC, I started to run tcpdump (and more recently snort) to get a feel for what was happening on some of our core servers and to see what Intrusion Detections programs I would like to start using. Trace B is from /var/adm/messages on dns-serv1.

3. Probability the source address was spoofed

What I am seeing is more likely a probing and/or attempted exploit of bind on our DNS machines, rather than one of the documented DOS exploits against name servers, so I feel it is unlikely the source addresses are being spoofed. The attackers would more likely be interested in being able to see the results of their scanning and would use a real source address.

4. Description of attack:

CVE numbers for BIND exploits:

CVE-1999-0833
CVE-1999-0009
Other related entries: CVE-1999-0835, CVE-1999-0848, CVE-1999-0849, CVE-1999-0851

The activity I have been seeing suggests that at least one person has been scanning our dns servers for the version of bind being run and possibly trying to break in. There are quite possibly multiple people probing these machines as I see both subtle OS detection type scans and brute force domain transfer attempts. Various 'Continuing exploits' of name servers are discussed at:

http://www.cert.org/advisories/CA-2000-03.html and http://www.cert.org/advisories/CA-99-14-bind.html

Evidence of possible exploit attempts can be seen in my traces. I initially started to look more carefully at our DNS servers after seeing some interesting items in /var/adm/messages on a couple of our DNS servers. As you can see from Trace B, there have been man zone transfer attempts, including at least one by an enterprising attacker, who attempted zone transfers from each of our subdomains.

More worrisome evidence is shown by the line:

```
Jul 29 14:28:56 dns-serv1 named[5692]: dropping source port zero packet from [216.x.x.x].0
```

This strikes me as a possible occurrence of the SF, source Port 0 scan or exploit signature. Unfortunately, these occurred at a time when I was not running tcpdump and out firewall administrator could not find any example of the probes. He has been moving our firewall to a much faster machine and there were some gaps in the logs. Another interesting snippet are the lines like:

```
Aug 06 14:40:12 dns-serv1 inetd[3626]: accept (for echo): Software caused connection abort
```

I have seen errors like this while playing with nmap, so it's possible someone was running nmap scans against some of our servers.

Trace A shows more worrisome evidence. The first part shows a BIND version query (One of the intrusion detection books from the SANS conference suggests BIND 8 and above does not respond to this query but I found that 8.2.2P5 does). I searched the Internet and found a program called nscan which scans for BIND, checks the version and tries an exploit if the version is exploitable and I saw a very similar result from that scan. Another thing I noticed was activity that confirmed the nmap scan. I didn't include the whole snippet, but you can see the nmap type strategy of trying various TCP flag combinations and so on.

5. Attack mechanism:

There are several common exploits in BIND that may being tried. One is a buffer overflow resulting from Bind doing an improper validation of NXT records. A second is a DOS exploit due to BIND doing an improper validation of SIG records. A third is due to a bug which allows attackers to cause BIND to consume more file descriptors than can be managed, causing named to crash. A fourth vulnerability is another denial of service which can be caused locally if certain permission conditions are met when validating zone information loaded from disk files. A fifth involves a problem with BIND not closing TCP sockets.

.See http://www.securityfocus.com/bid/866.html for more on all of these.

6. Correlations:

I just started recently to do more active Intrusion Detection on our DNS servers and our firewall administrator has been moving our firewall to a new machine, playing with the rule-sets, etc, so he has not been able to give me much in the way of logs. Correlating my short term tcpdump dumps and the /var/adm/messages files, however, points to active scanning and/or targeting of our DNS servers.

7. Evidence of active targeting:

Again, the tcpdump logs and /var/adm/messages logs point to some very active targeting. I see a very large number of zone transfer attempts, at least one nmap scan and attempts to query the version of BIND we are running, all evidence of active targeting.

8. Severity:

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

Criticality
5 (The host being attacked is a DNS server .)
Lethality
3 (Attacker at the very least probably has more information, such as the version of BIND we are running, to work with and may have actually tried exploits.)
System Countermeasures
3 (The host being attacked was running a newer OS with all patches but BIND has lots of potential exploits)
Network Countermeasures

2 (Being an education institution, it's difficult politically to have an aggressive firewall. We are making progress, but we have a fairly permissive firewall at this point.)

Severity = (5 + 4) - (4 + 2) = 3

No matter what the severity number, the signs point to a fairly high severity level. Our DNS machines are well patched and we try to keep BIND up to date, but we need to keep a better eye on these machines.

9. Defensive recommendation:

Start blocking more activity to our DNS servers via our firewall and start doing active Intrusion Detection, at very least for core servers.

10. Multiple choice question:

Question: Tcpdump log entries like this one:

```
21:56:56.236634 dialup.isp.com.1032 > dns-serv1.some.edu.domain:  5665 inv_q+ [b2&3=0x980] A? . (27)
21:56:56.238651 dns-serv1.some.edu.domain > dialup.isp.com.1032:  5665 inv_q Refused [0q] 1/0/0 (27)
21:56:56.238718 dialup.isp.com.1032 > dns-serv1.some.edu.domain:  64104+ [b2&3=0x180] TXT CHAOS)? version.bind. (30)
```

most likely:

A. Represents a DNS query resulting in no authoritative records
B. Represents an attempt to query the DNS server program for it's version number
C. Represent an attempted DNS Zone Transfer that failed
D. Represents an attempted port scan of a DNS server

Answer is B

3) Wuftpd 2.60 Buffer Overflow

Trace A – Email to root triggered by tcpwrappers

```
>From  server-maint-owner  Thu Jul 27 12:35:22 2000
Date: Thu, 27 Jul 2000 12:35:21 -0400
From: root@server2.some.edu
To: server-maint@some.edu
Subject: ALERT, ftpd@server2.some.edu accessed by
aaa1.cable.modem.be

 [aaa1.cable.modem.be]
```

Trace B – From the victims logs

```
> Aug  7 20:30:14 victim1-fw.138 netlogger: TCP cracked.linuxbox.some.edu:1332 ->
```

```
scanned.host.se:21 flags S destination iface eth1: Denied
> Aug  7 20:30:14 victim1-fw.138 netlogger: TCP cracked.linuxbox.some.edu:1333 ->
scanned.host.se:21 flags S destination iface eth1: Denied
```

Trace C – From compromised host

```
Jul 27 16:44:58 cracked ftpd[1167]: ANONYMOUS FTP LOGIN FROM aaa1.cable.modem.be [192.xxx.xxx.xxx],
guest@here.com
Jul 27 16:45:17 cracked ftpd[1167]: FTP session closed
```

Trace D – From the .edu Firewall logs

```
20000727 12:35:15.126880   60 T [14]RSTACK 00000000 247285C2
aaa1.cable.modem.be:ident -> server1.some.edu:33626
>20000727 12:35:19.163285   60 T [14]RSTACK 00000000 563EA0F0
aaa1.cable.modem.be:ident -> server2.some.edu:34141
>20000727 12:35:19.583141   60 T [14]RSTACK 00000000 FCB235DA
aaa1.cable.modem.be:ident -> server3.some.edu:34322
>20000727 12:35:19.944126   60 T [14]RSTACK 00000000 7140924A
aaa1.cable.modem.be:finger -> server3.some.edu:34142
>20000727 12:35:21.017363   60 T [14]RSTACK 00000000 9C797363
aaa1.cable.modem.be:ident -> server4.some.edu:34442
…
>20000727 12:45:30.751097   60 T [14]RSTACK 00000000 3759AD1F
aaa1.cable.modem.be:ident -> server9.some.edu:33495
>20000727 12:45:31.068957   60 T [14]RSTACK 00000000 D08415CE
aaa1.cable.modem.be:finger -> server9.some.edu:33498
```

1. Source of trace

I have included several traces or log entries so that I can show the correlation between host logs, the .edu firewall logs and the victim's firewall logs. Traces A, C & D are from the .edu network I am basing my detects on and Trace B was sent to me by one of the victims.

2. Detect was generated by:

Trace A is an example of email messages that Systems Administrators in my group get when someone tries to connect to any of the common services (ftp, rshd, rexecd, etc) on our core servers. As I indicated above, we use tcp-wrappers on many of our unix hosts. Tcp-wrappers is a small program written in C which allows connections to services started from inetd on unix systems to be logged and to trigger shell scripts. We use a script which logs the event and attempts to finger the host that the connection came from.

Trace B is a sample from the victim. It is a sample from their firewall showing the attempted ftp connections.

Trace C is a snippet from /var/log/messages on the compromised linux host. It shows the initial, anonymous login from the attacking host. Unfortunately, the log is missing everything after this event. I later discovered root's .bash_history evidence that the attacker had edited the messages files, presumably eliminating subsequent entries.

Trace D is from the .edu firewall logs, showing the attacker's scan of hosts that are running ftp servers, or at least respond as if they were.

3. Probability the source address was spoofed

The source address of the attacker was not spoofed. The attacker initially scanned our address space for vulnerable ftp servers, which means they wanted the information back and wouldn't use a spoofed address, and subsequently logged into compromised host from that source address.

4. Description of attack:

The attack was one of the well documented exploits of wuftpd. This particular attack took advantage of a problem with the SITE EXEC command in wuftpd 2.6.0 running on a Redhat Linux host. I found information on this attack at: http://www.ciac.org/ciac/bulletins/k-054.shtml

Attack CVE: CVE-1999-0080

We discovered the attack while investigating complaints from a couple of European sites indicating that their networks had been scanned by host within our domain. After receiving the complaints, I contacted the administrator of the host. He and I took a look at the host and quickly realized it had been compromised. Accounts had been added which the administrator new nothing about, /bin/login has been replaced and account names and passwords were being logged to a file as users logged in, and I found various hacker tools. We immediately took the compromised machine off the network and I started doing some offline forensics. I won't go into every detail, but I did notice two pertinent things: 1) I found two binary programs, one which was called wu-scan and another called wu, and a log of hosts in a European domain that were presumably running vulnerable ftp servers. I was not able to find the source code for the programs, but running the strings command on each gave me the usage statements and this was enough information for me to search the Internet and locate the tools.

Strings on wu-scan produced:

```
wu-scan by Narrow (nss@privacyx.com)
Usage: %s <IP List>
```

Strings on wus produced:

```
Usage: %s -t <target> [-l user/pass] [-s systype] [-o offset] [-g] [-h] [-x]
        [-m magic_str] [-r ret_addr] [-P padding] [-p pass_addr] [-M dir]
target    : host with any wuftpd
user      : anonymous user
dir       : if not anonymous user, you need to have writeable directory
magic_str : magic string (see exploit description)
-g        : enables magic string digging
-x        : enables test mode
pass_addr : pointer to setproctitle argument
ret_addr  : this is pointer to shellcode
```

wu-scan is a program which searches a list of IP's (the hacker appeared to scan domains for hosts running any ftpd service to produce this list) and checks them to see if they are running a vulnerable copy of wu-ftpd. Wus appears to be a program which actually exploits the vulnerable hosts using the SITE EXEC command (confirmed by the source code)

2) The second thing I noticed was that source address of the anonymous ftp connection in /var/log/messages. I recognized the name from one of the tcp-wrapper email messages we had received about probes of the ftp service on one of our servers. This gave me an important piece of information for correlating these logs with information in our firewall logs (see correlation section below)

5. Attack mechanism:

wuftpd is a ftp server. Due to insufficient checking in the formatting of the "site exec" command, it is possible to coerce the wu-ftpd daemon to execute arbitrary code. This vulnerability may allow local, remote and anonymous users to gain root privileges.

Given this, the attacker locates hosts running vulnerable versions of wuftpd and then runs a program which logs in anonymously to this server and coerces the ftpd daemon to give the attacker a root shell.

6. Correlations:

With the information I gained from the compromised host, I was able to get a fairly good picture of the sequence of events. I asked our firewall administrator to look for connections from the host which had anonymously logged into the compromised machine and who had triggered the tcpwrappers emails and he was found a series of connections to machines on our network running ftp servers. Trace D shows a snippet of the results. The trace shows SYN-ACKs back from several of our core file servers . The log shows the date, size of packet, protocol (T for TCP, U for UDP), the TCP flag settings, sequence and ack numbers.

```
20000727 12:35:15.126880   60 T [14]RSTACK 00000000 247285C2
aaa1.cable.modem.be:ident -> server1.some.edu:33626
```

These servers all log the connection and then reset the connection using tcp-wrappers. You can see from the trace that they also attempt to run ident on the foreign host as well as try and finger the host.

7. Evidence of active targeting:

There is very strong evidence of active targeting, by at least one person. Given the evidence I found on the compromised host, the attackers appears to 1) Scan networks for hosts running vulnerable ftp servers and produce a list of those hosts and 2) Run a program which can exploit these hosts using the list of vulnerable hosts as input. He or she then places trojan horses on the compromised hosts (like replacing /bin/login) to capture passwords and so on. The host is then used to scan another network and the attacker can leap frog from network to network, building up a collection of compromised hosts.

8. Severity:

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

Criticality
3 (The compromised host is a linux server but it part of a NFS cluster which share common accounts and passwords.)
Lethality
4 (The attack succeeded and the attacker was logging passwords and possibly sniffing the network for more. The attacker could potentially have broken into other systems over time. I might even rate this a 5 except that our core servers only allow ssh connections and these systems were on a subnet with low privilege users; no campus wide administrators passwords could have been captured.)
System Countermeasures

2 (The compromised host was running a newer OS, Redhat Linux 6.2, but such a system is fairly vulnerable unless very well patched and the administrators of this machine had not applied any of the patches to 6.2 at all.)

Network Countermeasures

2 (Being an education institution, it's difficult politically to have an aggressive firewall. We are making progress, but we have a fairly permissive firewall at this point.)

Severity = (3 + 4) - (2 + 2) = 3

The calculation shows a severity of 3 but I might even make it a 4. The exploit was a targeted one and compromised a non core system. This allows the attacker to gather passwords, to launch attacks against other sites, which is damaging to our reputation, and gives them a better chance at getting foothold into one of our core systems.

9. Defensive recommendation:

There are many possible defensive strategies to come out of this. For a compromised host, not matter what the exploit, we remove that host from the network until the host is rebuilt (OS re-installed) and all current patches applied. The host in this case was a unix/linux hosts, so we work with the administrator of the host to remove any unnecessary services (sendmail, bind, NFS daemons, etc) from the host, comment out everything we can from inetd.conf, install or start using tcp-wrappers and host intrusion detections systems like portsentry. We also start looking at connections to this host more actively, at least for the short term, and often run scans of the machines over time to make sure that the users don't simply start up unnecessary services again.

On a network scale, we are looking at possibly restructuring our subnets so that we have subnets for servers and subnets for individual PCs and workstations. The workstation subnets could then be restricted via the firewall so that no external access other than services like ssh would be allowed. This would follow our general strategy of structuring our network to prevent many exploits from even being possible or at least beyond the scope of novice attackers and would allow us to simplify our firewall rulesets, which is important since we are moving to faster and faster connections to the Internet.

10. Multiple choice question:

Question: Many exploits involve the use of Buffer Overflows. Buffer Overflows are:

A. Denial of Services attacks similar to SYN Flooding, where the TCP Buffer of the victim is filled, preventing new packets from being processed.
B. A result of a problem with the way a program or Operating System is written, allowing arbitrary or unexpected code to be run by the attacker, leading to a Denial of Service or higher privileged access to the system for the attacker.
C. A mechanism for fragmenting packets such that the fragment offsets overlap, causing the buffer which a host uses to re-assemble fragments to fill.
D. Similar to Ping of Death attacks, where very large ICMP packets are sent to a host, using up it's memory and causing the machine to crash.

Answer is B

4) Trojan Horse Scanning

<u>Trace A – BlackIce Defender log</u>

```
59         2000-07-22    20:28:49    2003103     NetBus     probe      35.x.x.x      KRAFT     192.168.24.228
port=12345&name=NetBus 1
```

Trace B – some.edu Firewall log

```
20000722  11:54:47.979560   60  T  [04]RST    B2B79860  00000000  209.x.x.x:48228  ->  192.168.161.125:12345
DFBIT
20000722  11:54:47.981763   60  T  [04]RST    B2C95F6F  00000000  209.x.x.x:48249  ->  192.168.161.125:12345
DFBIT
20000722  11:56:01.700027   60  T  [04]RST    BECE40E3  00000000  209.x.x.x:51193  ->  192.168.161.125:12345
DFBIT
20000722  11:56:01.703015   60  T  [04]RST    BED8CBE2  00000000  209.x.x.x:51205  ->  192.168.161.125:12345
DFBIT
```

1. Source of trace

Both traces are from the .edu network I am basing my detects on.

2. Detect was generated by:

Trace A is from a Black Ice Defender log running on my staff PC. The second trace is from our firewall logs.

3. Probability the source address was spoofed

Unlikely. As the Black Ice Defender trace shows, these were probes for hosts running the Netbus Trojan.

4. Description of attack:

Netbus is a PC Trojan similar to Back Orifice. It allows a remote user to access and control your machine by way of its Internet link. NetBus was written by a Swedish programmer, Carl-Fredrik Neikter, in March 1998. Version 1.5 in English appeared in April. NetBus apparently received little media attention but it was in fairly wide use by the time BO was released on 3 August. One big difference between the original Back Orifice and Netbus is that Netbus ran on Windows NT while Netbus only ran on Windows 95/98.

5. Attack mechanism:

Netbus is a PC Trojan. The unsuspecting victim runs the program either directly or by way of a application used as camouflage, and it immediately installs itself and begins to offer access to attackers.

NetBus and other remote-admin trojans have two essential parts; a server (the part that resides on the victim's system) and a client (the application used to find and control the server). Features and functions vary, but the result is much the same; a remote attacker can control the victims computer. Once in place, these trojans open the victim to endless possibilities ranging from mere pranks to viruses, serious loss or theft of valuable or sensitive data, other trojans, and so on.

Netbus, like many other Trojans, runs at a standard port of 12345, so enterprising attackers can probe PCs for services running on this port to find Netbus victims.

6. Correlations:

There was a lot of correlating evidence indicating a Netbus probe of campus PCs. The firewall logs showed a large number of attempted connections from the Internet to port 12345 on campus hosts and all of the computers running Black Ice Defender I have access to indicated netbus probes.

7. Evidence of active targeting:

There was evidence of active targeting. The firewall logs show a large number of hosts being specifically scanned on port 12345, during a short span of time, which is not likely a random occurrence.

8. Severity:

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

Criticality
2 (The host(s) being attacked/probed are PC desktops .)
Lethality
2 (The most damage an attacker could typically do to a PC in our environment is securing private files, DOS attacks, etc.)
System Countermeasures
3 (The host being attacked was running a newer OS with some patches missing)
Network Countermeasures
2 (Being an education institution, it's difficult politically to have an aggressive firewall. We are making progress, but we have a fairly permissive firewall at this point.)

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

In our environment, the –1 is probably a valid Severity level. PCs running Windows are single user machines, which makes it more difficult for attackers to use them sniff passwords, break our core servers and so on. Also, the one group of subnets our firewall aggressively limits traffic to, our residential networks, are the most likely victims of this Trojans (we only allow external access to Port 80). The damage could be more severe if the PC involved were in one of our labs, was a NT Domain Controller or was a Windows 2000 Active Directory Server, but these are less likely victims. Our PC lab machines are automatically rebuilt nightly and our NT and Windows 2000 servers are well patched and Administrators don't generally install programs off the Internet on them. So, in general, the worst scenario would involve a loss of personal files and DOS attacks.

9. Defensive recommendation:

Block connections to all ports which PC Trojans at our firewall.

Question If you see scans or attempted connections to port 12345 on a PC, the attacker is probably looking for:

A. A Tribal Flood Network Master program
B. The Netbus Trojan
C. A WINS server
D. A PC Anywhere client

Answer is B

5) ICMP Flooding

http://www.cert.org/advisories/ CA-98.01.smurf.html

```
20000720 09:38:05.625269 74 I a1.sj.cable.modem.com ->c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.632448 74 I cm-aa-bb-cc-dd.newcable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.643485 74 I cm-24-29-74-48. newcable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.645635 74 I sc-24-30-167-109.ny.cable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.647880 74 I avg-27-184-154.cinci.cable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.651917 74 I avg-27-184-154.cinci.cable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.653415 74 I aa.cc.d.e.another.cable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.657277 74 I cvg-27-184-154.cinci.cable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.658948 74 I we-24-31-53-142.we.avg-cable.net -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.659151 74 I nic-21-c12.mn. avg-cable.net -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.661668 74 I nic-25-b17.mn. avg-cable.net -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.662303 74 I avg-27-184-154.cinci.rr.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.666405 74 I avg-27-184-154.cinci.rr.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.669198 74 I nic-25-c.mn. cableisp.net -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.676833 74 I nic-25-c.mn.cableisp.net -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.677620 74 I cr101.bc.ca-cable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.685689 74 I cm-24.ny.cable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.691152 74 I cm-74.ny.cable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.697911 74 I cs276-67.austin.southern.cable.com -> c1.dynamic.some.edu
Type:ECHOREPLY RATELIMITED
20000720 09:38:05.717801 74 I ros75-155.not-a-cable.net -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.814116 74 I cm-157-51.ny.modem.cable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.863465 74 I cs28133-216.satx.justcable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.870352 74 I csdo-21179.anycom.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.873988 74 I ubr-27.apop.cfl.cable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.937293 74 I cs28133-216.cable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.954131 74 I 24.68.on.cablehome.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
20000720 09:38:05.954569 74 I cs281.satx.cable.com -> c1.dynamic.some.edu Type:ECHOREPLY
RATELIMITED
```

```
20000720 09:38:05.955811  74  a.crdva1.bc..homecable.com  ->  c1.dynamic.some.edu  Type:ECHOREPLY
RATELIMITED
```

1. Source of trace

Firewall logs from the .edu network I am basing my detects on.

2. Detect was generated by:

Our custom written firewall.

3. Probability the source address was spoofed

Very likely. This traffic went on for hours and we initially monitored the traffic between the various hosts and c1.dynamic.some.edu. All of the echo replies were unsolicited (NO echo requests or echo replies from c1) but the source address was c1.dynamic.some.edu.

4. Description of attack:

This was clearly ICMP flooding via spoofing the source address and could have been a variation on the smurf attack or possibly a Tribal Flood network attack. The trace shows ICMP echo reply packets coming from many sources to one some.edu host. In each line, you see the date, the time, the size of the packet, the host sending the ICMP echo reply, the destination host, the ICMP Type, which is an Echo Reply, and the information field indicating the firewall is rate limiting ICMP echo replies.

5. Attack mechanism:

The attacker seems to have send icmp echo requests to a large number of cable land hosts with the (spoofed) source address of c1.dynamic.some.edu. I only show a portion of the log above; Overall, over fifty hosts were sending echo replies to c1.dynamic.some.edu. This looked to me like a variation on the Smurf attack, except that rather than sending the echo requests to a network broadcast address (which may be more difficult now as sites block this), they used a large number of individual hosts (mostly cable modem, probably because many of these are left on all the time).

6. Correlations:

I did not spend a lot of time correlating this particular attack. Our firewall keeps some state and it starts to rate limit ICMP packets over a certain threshold, so there is little impact on our network. It's possible that a larger attack could cause the firewall or router problems, but this attack was not large enough to cause problems.

7. Evidence of active targeting:

There is strong evidence of active targeting with this attack. The relatively large number of hosts, all sending replies in a small time frame to one host, using a spoofed source address, suggests that someone was actively trying a DOS attack. Plus, these packets were obviously crafted.

8. Severity:

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

Criticality
2 (The host being attacked is a Unix desktop .)
Lethality

2 (Without the rate limiting our firewall does, the attack could cause the host problems.)

System Countermeasures

3 (The host being attacked was running a newer OS with some patches missing)

Network Countermeasures

2 (Being an education institution, it's difficult politically to have an aggressive firewall. We are making progress, but we have a fairly permissive firewall at this point.)

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

In our environment, the -1 is probably a valid Severity level. If the number of hosts involved had been higher and our firewall didn't rate limit the packets, it may have caused the host some problems and been an larger inconvenience but it would still be fairly low in severity.

9. Defensive recommendation:

Block as much ICMP traffic at your firewall as your users will stand or rate limit ICMP replies.

10. Multiple choice question:

Question: An unsolicited ICMP echo reply could be explained by:

A. Smurf Attack

B. Communication between a TFN Master and Daemon

C. A covert Loki channel

D. All of the above

Answer is D

Assignment Two:

The attack I chose is a buffer overflow in rcp.statd on linux systems. I got the attack from:

http://www.securityfocus.com/bid/866.html

CVE-1999-0184

Bugtraq ID 1480 - Multiple Linux Vendor rpc.statd Remote Format String Vulnerability

A vulnerability exists in the rpc.statd program which is part of the nfs-utils package, distributed with a number of Linux distributions. Because of a format string vulnerability when calling the syslog() function a malicious remote user can execute code as root.

The rpc.statd server is an RPC server and is one component of the Network File System (NFS) architecture; It is used by the NFS file locking service. It was introduced because NFS is stateless, but file locks are inherently stateful. So the problem arises how to deal with locks if either the server or a client crashes. In that case, the crashed box, when it comes up again, sends out notifications via NSM to statd's running on its NFS clients/servers[*]. That statd has been configured to call back the local NFS file locking daemon (lockd) so that it can release stale locks and/or reclaim previously held locks.

The logging code in rpc.statd uses the syslog() function, passing the data using a user supplied format string. A malicious user can construct a format string that injects executable code into the process address space and overwrites a function's return address, thus forcing the program to execute the code. rpc.statd requires root privileges for opening its network socket, but fails to drop these privileges later on. so code executed by the malicious user can execute with root privileges. Any Linux distribution which runs the statd process is apparently vulnerable, unless patched for the problem.

I located several programs at http://www.securityfocus.com which presumably could exploit the vulnerability. I chose one. Written by a person who called themselves doing, and ran it in a controlled environment. I am very interested in linux exploits because of the wuftpd breakin I encountered (see Detects above). We are starting to see linux clusters which use NFS for allowing common access to data, and I want to be able to give people ways to do this in a more secure manner.

I initially compiled the program as is and tried using the stock buffer offsets it supplied to break into a standard Redhat 6.2 host. As indicated above, this exploit works by constructing a format string for the syslog function that injects executable code into the process address space. The complication to this is that, for the exploit to work, the correct address space offset must be known for the particular host being attacked. The correct offset varies between linux distributions but the program came with offsets for various Redhat distributions. Unfortunately (for my tests at least), I could not get the default offset to work. When I ran the program, I would get these results:

```
 tcpdump: listening on eth0
00:18:00.755669 myhost.is.some.edu.799 > target.host.some.edu.sunrpc:  udp 56
00:18:00.758622 target.host.some.edu.sunrpc myhost.is.some.edu.799:  udp 28
00:18:00.759252 myhost.is.some.edu.800 > target.host.some.edu.953:  udp 1076
00:18:00.764356 target.host.some.edu > myhost.is.some.edu: icmp: target.host.some.edu udp port 953
unreachable [tos 0xc0]
00:18:05.767085 myhost.is.some.edu.2606 > target.host.some.edu.39168: S 2145187954:2145187954(0) win
32120 <mss 1460,sackOK,timestamp 184585265[|tcp]> (DF)
00:18:05.768143 target.host.some.edu.39168 myhost.is.some.edu.2606: R 0:0(0) ack 2145187955 win 0
```

```
[root@myhost /tmp]# ./statdx -a 0x41414141 target.host.some.edu
buffer: 0x41414140 length: 999 (+str/+nul)
target: 0x41414544 new: 0x41414398 (offset: 600)
wiping 9 dwords
clnt_call(): RPC: Unable to receive; errno = Connection refused
A timeout was expected. Attempting connection to shell..Failed
```

I searched the Internet further and found that the author had actually posted various versions of the program in attempt to produce a program that worked in most cases. I finally found a series of comments by him at http://www2.merton.ox.ac.uk/~security/bugtraq-200008/0031.html, which explained how to get the correct offset and was able to compromise a machine.

Assignment Three:

This assignment involved looking at a months worth of snort alert logs and trying to figure out what was going on at the site. Given such a large amount of data, and what appeared to be a large amount of activity at this site, I decided to:

1. Look through the logs and identify all alerts which had been triggered, so I could first get a global picture of the types of alerts. Once I got a picture of the types of alerts, I could start looking for additional information based on some of the important things took look at when analyzing logs. These include:

   A. What is the protocol involved?
   B. If the protocol is TCP, what flags are set?
   C. Anything unusual about ID numbers, sequence number, ack numbers, fragment Ids, etc?
   D. Is fragmentation involved? Any negative offsets of funny business, small fragment sizes?
   E. What source and destination ports are involved? Are they ports related to common services? Known trojan horse ports? Is there a pattern to the ports?
   F. Are the TTLs tell us anything?
   G. Do the source and/or destination host tell me anything? Is there any pattern, are they spoofed, do the domains they come from indicate anything (like cable modem land)?
   H. Do the packets follow normal conventions (normal flags, offsets multiple of 8's, etc)?
   I. Is there any missing data, like missing packets or logs?
   J. Any scanning involved? Is it host or port scanning? Are they scanning UDP, TCP, etc, stealth or straight forward scans?
   K. What is the timing of events? Is every in a small time period or spread out?
   L. Was the TCP handshake completed?
   M. Are any attack or attackers signatures evident ?

2. Here are the alerts I found:

```
[**] Watchlist 000222 NET-NCFC [**]
[**] WinGate 8080 Attempt [**]
[**] WinGate 1080 Attempt [**]
[**] GIAC 000218 VA-CIRT port 34555 [**]
[**] Null scan! [**]
[**] SNMP public access [**]
[**] Watchlist 000220 IL-ISDNNET-990517 [**]
[**] SMB Name Wildcard [**]
[**] Tiny Fragments - Possible Hostile Activity [**]
[**] Probable NMAP fingerprint attempt [**]
[**] GIAC 000218 VA-CIRT port 35555 [**]
[**] SUNRPC highport access! [**]
[**] spp_portscan: PORTSCAN DETECTED from 212.240.94.225 (STEALTH) [**]
[**] SYN-FIN scan! [**]
[**] GIAC 08-feb-2000 [**]
[**] Happy 99 Virus [**]
[**] NMAP TCP ping! [**]
```

3. The next thing I did was to figure out what each alert was triggered by

```
[**] WinGate 1080 Attempt [**]
[**] WinGate 8080 Attempt [**]
```

WinGate allows the simultaneous sharing of an Internet connection with a computer network. It basically allows a Windows98 computer to act as a proxy. Unfortunately, the default configuration is too permissive and allows anyone to use this computer to connect anywhere, thus hiding his real IP address. WinGate server often don't have a password set, and thus can be used by crackers from anywhere as a telnet relay.

[**] Null scan! [**]

This alert indicates a TCP packet with no flags set. These are crafted packets designed to evade defenses and possibly to do OS identification.

[**] SNMP public access [**]

An alert to look for SNMP community string, probably content based

[**] Watchlist 000220 IL-ISDNNET-990517 [**]
[**] Watchlist 000222 NET-NCFC [**]

The watchlist rules I would guess are for alerting the analyst to traffic to/from networks of interest to the analyst. I would guess the IL ISDNNET refers to an network in Illinois with ISDN connections, for example. I see no direct mention of it in the SANS Activity Alerts for Feb 20th 2000, but the 000220 in the Alert would indicate to me that something in the report triggered their interest. The watching of NCFC, which looks like a Chinese network, was triggered by a SAN Activity trace elicited on Feb 22nd 2000.

[**] SMB Name Wildcard [**]

This alert refers to IDS177 , which indicates a standard netbios name table retrieval query.

Windows machines often exchange these queries as a part of the filesharing protocol to determine NetBIOS names when only IP addresses are known. An attacker could use this same query to extract useful information such as workstation name, domain, and users currently logged in.

[**] Tiny Fragments - Possible Hostile Activity [**]

The alert detected tiny fragments, which is quite possibly hostile in nature.

[**] Probable NMAP fingerprint attempt [**]

The alert indicates a (probable) nmap OS fingerprint scan.

[**] GIAC 000218 VA-CIRT port 34555 [**]
[**] GIAC 000218 VA-CIRT port 35555 [**]

This alert refers to a detect posted to the GIAC Activity Reports on Feb 18th 2000. The person posting the detect indicated they had found 16 Windows 98 computers infected with BackOrifice which were involved in UDP flood attacks. They noticed a process called service.exe running on these machines and running strings on the exe resulted in the string trinoo. The machines were all found to be sending UDP packets from port 35555 and receiving packets on port 34555.

[**] SUNRPC highport access! [**]

An access of one of the (high ports) typically used by Sun RPCs, port 32771, probably to exploit a Solaris rpcbind vulnerability. The Black Ice Defender exploits lists says:

Ghost Portmapper - Some SunOS machines listen at this port for portmapper. Since firewalls frequently don't filter at high ports, it can allow the attacker access to portmapper even when port 111 is blocked.

See http://advice.networkice.com/advice/Exploits/Ports/32771/default.htm

[**] spp_portscan: PORTSCAN DETECTED from 212.240.94.225 (STEALTH) [**]

spp_portscan indicates the use of Patrick Mullen's Snort Portscan Preprocessor from http://spyjurenet.com/linuxrc.org/projects/snort/

This gives snort the ability to look for scans, stealth scans, nmap scans, etx.

[**] SYN-FIN scan! [**]

Indicates a scan with the SYN and FIN TCP flags set, possibly a stealth probe

[**] GIAC 08-feb-2000 [**]

This rule looks for source traffic from 195.11.50.204. The title refers to the fact that one of the traces posted to GIAC on Feb 8th 2000 indicated correlation to the source address from another posting made Feb 5th 2000. This is the beauty of posting traces to a common area or site. Multiple posters can then correlate what they see and others can develop rules to look for addresses such as these to correlate activity further. Knowing that others have seen this source address may allow you to see significance which may otherwise be overlooked.

[**] Happy 99 Virus [**]

Happy 99 is a Win32 based Trojan program. When this program is executed, it displays fireworks. Apart from the fireworks, this program will allow for other activity in the background without the user's permission, such as creating two files SKA.EXE and SKA.DLL. It will alter WSOCK32.DLL to put its code into that file and keep the original file as WSOCK32.SKA. It can not modify the WSOCK32.DLL file if it is in use. In such a case, this program will add an entry to the Windows Registry to run SKA.EXE the next time the computer is booted so that it can then perform such modifications. The size of this Trojan file is 10,000 bytes. A user has to actually execute Happy99 to become infected. The modified WSOCK32.DLL has routines to detect the email and newsgroup postings made by the user. It will send a copy of the SKA.EXE file renamed as happy99.exe to every user or newsgroup to whom the user has sent an email, helping it infect other users.

[**] NMAP TCP ping! [**]

A NMAP TCP 'Ping' is used to check the availability of a system without sending ICMP echo requests, which may be blocked by some sites. A TCP "ping" will send an ACK to each machine on a target network. Machines that are up should respond with a TCP RST

2. The next thing I did was look for anything unusual about the logs themselves

   I noticed that SnortS10.txt is very small (6/10) and the data truncated, so something must have happened on this day.

3. The last thing I did was analyze the log based on the alerts

[**] NMAP TCP ping! [**]

On 5/28 and 5/29, all of MY.NET.16 and MY.NET.19 were scanned. On 5/31 and 6/1 , MY.NET.101 and MY.NET.102 were scanned. Each host was probed at 3-5 ports and there was roughly a three minute difference between each host being scanned. All the scans were from MY.NET.253.12:43758.

The analysts probably caught part of a scan of the entire MY.NET, being done somewhat stealthily (a subnet a day, host scans 3 minutes a part, subnets not in order. I would guess that MY.NET.253.12 is compromised. Either way, the machine and who uses it should be looked at.

[**] Happy 99 Virus [**]

I only found three Happy 99 scans:

```
SnortA19.txt:06/13-10:26:37.292191  [**] Happy 99 Virus [**] 207.172.132.67:1038 -> MY.NET.253.52:25
SnortA5.txt:05/25-09:53:44.364111  [**] Happy 99 Virus [**] 207.172.145.30:1294 -> MY.NET.253.51:25
SnortA8.txt:05/25-09:53:44.364111  [**] Happy 99 Virus [**] 207.172.145.30:1294 -> MY.NET.253.51:25
```

The Happy 99 Virus alert is content based. The destination host port is the SMTP port. I would guess the Happy 99 virus is being sent via email.

[**] Null scan! [**]

Multiple hosts are doing null scans of subnets. Some of them are from our friend MY.NET.253.12, who was doing the NMAP TCP pings. The subnets being scanned by this person are the same ones on the same days as the NMAP scans, so they are probably a result of the NMAP scan.

[**] SUNRPC high port access! [**]

Multiple hosts are doing SUNRPC scans of subnets. Some of them are from our friend MY.NET.253.12, who was doing the NMAP TCP pings. The subnets being scanned by this person are the same ones on the same days as the NMAP scans, so they are probably a result of the NMAP scan. A lot f the other traffic may be normal traffic. I see:

```
28.8.10.141:23 -> MY.NET.2.203:32771
```

for example, but this looks like a normal telnet session where the client just happened to pick port 32771, which triggered the alert but is explainable. I also see what are probably X connections which also probably is normal.

[**] GIAC 08-feb-2000 [**]

Essentially all of the activity triggering this alert is to one machine, MY.NET.179.77. It looks like a port scan of the machine.

[**] SNMP public access [**]

The source address for all of the events triggering this are MY.NET.97 and the destination address is always on MY.NET.101. Maybe someone setting up ucd-snmp on a linux box?

[**] SMB Name Wildcard [**]

I found a lot of traffic triggering this alert to MY.NET.101.192 and MY.NET.100.130

[**] WinGate 1080 Attempt [**]
[**] WinGate 8080 Attempt [**]

I found a lot of Wingate 8080 traffic to MY.NET.253.105, MY.NET.99.85 and MY.NET.97.69, from host all over the place. I would guess these are hosts running Wingate as a proxy with no password and are being used to proxy telnet sessions or such. I found a lot of Wingate 1080 traffic to various host on MY.NET. Individual destination hosts stood out here, so this could be illegal proxying, but is probably more likely normal usage of Wingate. I also found our good friend MY.NET.253.12 scanning hosts for Wingate 1080 and 8080 on MY.NET.16. This was, again, probably part of their NMAP scan.

[**] Watchlist 000220 IL-ISDNNET-990517 [**]

I think this is just a subnet they analysts are interested in monitoring. Nothing jumps out at me, although I see what appears to be IRC traffic, some gaming activity possible, a ftp server, etc. This may simply be similar in nature to a cable network (gaming, IRC, etc). The analysts normal pattern for their Alert Messages would indicate that a 2/20 SANS Activity Detect triggered their interest in watching this net/subnet. I see a lot of detects showing activity from 212.25, so this may be what interested them in looking at traffic from 212.179.x.x. I also see a SANS Activity entry from March 30th 2000 with this same alert message.

[**] Tiny Fragments - Possible Hostile Activity [**]

Almost all of this is between these hosts:

206.193.209.254 (userproxy.inacom.com) -> MY.NET.219.58 and these hosts:
24.3.7.221 (a cable modem) -> MY.NET.70.121

I also see some SYNFIN probing of one of these destination hosts:

208.220.120.13:53 -> MY.NET.219.58:53 SYNFIN **SF****

I don't see any port numbers, so I assume the traffic is ICMP tiny fragments. A covert channel perhaps?

[**] SYN-FIN scan! [**]

I see what looks like a high speed SYNFYN scan from 204.60.176.2, port 53 of the entire MY.NET network, going in sequence through each subnet and each host within a subnet, to port 53. Looks like they are looking for DNS servers, using a SYNFIN stealth approach.

[**] Watchlist 000222 NET-NCFC [**]

I see lots of traffic which indicates hosts like MY.NET.100.230 and MY.NET.6.7 are talking to the SMTP port on various hosts, most of them on the 159.225 network. It looks like host makes a SMTP connection to a host on the 159.225 network and answer back from the SMTP server triggers the alert. I also see SMTP traffic to MY.NET.253.43. There are a few unusual things, like I see connections from 159.226.159.1 to the SMTP port on MY.NET.253.43, MY.NET.253.42 and MY.NET.253.41. The GIAC Activity page for 2/22 infers this network is connected to The Computer Network Center Chinese Academy of Sciences, China, and shows some funny portmapper connections, so I would guess this is why the analysts are keeping track of it.

```
[**] GIAC 000218 VA-CIRT port 34555 [**]
[**] GIAC 000218 VA-CIRT port 35555 [**]
```

I see activity to ports 34555 and 35555 on MY.NET.253.x, but it looks like these were clients connecting to legitimate services like mail, dns and ident, where the clients happen to have chosen a port which triggered the alert.

```
[**] spp_portscan: PORTSCAN DETECTED from 212.240.94.225 (STEALTH) [**]
```

I see a lot of activity from our friend MY.NET.253.12, again, probably an NMAP scan.

Conclusion

The most worrisome activities I see are the NMAP scans from MY.NET.253.12, which may be compromised, the seemingly open WINGATE 8080 PROXY servers, the small fragment activity and the SMB Wildcards. I would immediately cut off network access to MY.NET.253.12 and investigate it first.