Global Information Assurance Certification Paper

Intrusion Detection In Depth

GCIA Practical Assignment

Version 3.0

SANS Darling Harbour, Sydney 2002

by

David Begg

25th March, 2002

Table of Contents

Assignment 1 – Describe the State of Intrusion Detection

Dealing With Windows RootKits

Introduction

This paper deals with a rootkit that is currently under development for Microsoft Windows NT and 2000. The testing was performed with NT Rootkit v0.40 [1] and Snort version 1.8.3 for Windows [2]

Background

Generally, the aim of attackers is to gain administrative (or "root" for Unix systems) control of any computer systems they can access. Once they have gained control, they want to ensure they can maintain that control.

There may be a number of reasons why an attacker wants ongoing control of a computer. They may want to get their hands on the valuable data the organisation has worked hard to produce. Maybe they want a safe hiding place for their hacking tools or other "warez". Or, they may be looking for a launching pad from which they can initiate attacks on other networks with minimum risk of being detected. It simply may be a way of proving their skill level to their peers. Any of these scenarios can potentially raise serious legal and financial implications for an organisation if it's system is left unsecured.

Once the attacker has gone compromised your computer, he doesn't want to have to do it again every time he accesses it. Compromising a computer may be hard work and there is a high risk of getting caught in the process. So, the attacker will probably install a "backdoor", a means of getting back in again with minimum effort and minimum chance of getting caught in the act. The commonly used types of backdoors are "Trojans " and "rootkits".

Even if the original vulnerabiliy they used is later plugged, the attacker will still be able to gain access until the backdoor is discovered and closed. In fact, a common procedure for attackers is to actually close the backdoor they ori ginally used to gain access to the host. This will make sure the next attacker who comes along does not evict them.

To date, rootkits are the most powerful backdoors available to hackers. The aim of this paper is to investigate how the existence of a windo ws rootkit can be detected using network based intrusion detection systems and what can be done to minimise vulnerability.

What is a Rootkit?

Traditionally, rootkits were the domain of Unix systems, but currently there is a rootkit under development for Wi ndows systems. As of this paper being written, the latest version of NT Root in development was 0.44, however the web sites containing this version were not available. The most recent version available on the Internet at the time of writing this paper was 0.40, obtained from the site listed under "References" at the end of the paper.

A rootkit does not exploit an existing vulnerability, but actually creates new vulnerabilities on a system. The attacker can easily use these in the future. Neither are rootkits tools that allow an attacker to gain administrator access. Rather, once

the attacker has gained administrator access, the rootkit lets him easily regain it whenever he wants.

Whatis.com defines a rootkit as:

"A rootkit is a collection of tools (programs) that a hacker uses to mask intrusion and obtain administrator -level access to a computer or computer network. The intruder installs a rootkit on a computer after first obtaining user -level access, either by exploiting a known vulnerability or cracking a password. The rootkit then collects userids and passwords to other machines on the network, thus giving the hacker root or privileged access.

A rootkit may consist of utilities that also: monitor traffic and keystrokes; create a "backdoor" into the system for the hacker's use; alter log files; attack other machines on the network; and alter existing system tools to circumvent detection.

The presence of a rootkit on a network was first documented in the early 90s. At that time Sun and Linux operating syste ms were the primary targets for a hacker looking to install a rootkit. Today, rootkits are available for a number of operating systems and are increasingly difficult to detect on any network." [3]

According to an article written by Greg Hoglund and published in Phrack Magazine:

"A rootkit is a set of programs which 'PATCH' and 'TROJAN' existing execution paths within the system. This process violates the 'INTEGRITY' of the TRUSTED COMPUTING BASE (TCB). In other words, a rootkit is something which inserts backdoors into existing programs, and patches or breaks the existing security system.

- *A rootkit may disable auditing when a certain user is logged on.*
- *A rootkit could allow anyone to log in if a certain "backdoor" password is used.*
- *A rootkit could patch the ker nel itself, allowing anyone to run privileged code if they use a special filename.*

The possibilities are endless, but the point is that the 'rootkit' involves itself in pre -existing architecture, so that it goes un -noticed. A remote administration applicat ion such as PC Anywhere is exactly that, an application. A rootkit, on the other hand, patches the already existing paths within the target operating system."

The same article also listed some potential exploits of an NT rootkit:

"*1. Insert invalid data. Invalid data can be inserted into any network stream. It can also introduce errors into the fixed storage system, perhaps subtly over time, such that even the backups get corrupted. This violates reliability & integrity.*

2. Patch incoming ICMP. Using ICM P as a covert channel, the patch can read ICMP packets coming into the kernel for embedded commands.

3. Patch incoming ethernet. It can act as a sniffer, but without all of the driver components. If it has patched the ethernet, then it can also stream da ta in/out of the network. It can sniff crypto keys.

4. Patch existing DLL's, such as wininet.dll, capturing important data.

5. Patch the IDS system. It can patch a program such as Tripwire or RealSecure to violate its integrity, rendering the program un able to detect the nastiness...

6. Patch the auditing system, i.e., event log, to ignore certain event log messages." [4]

While there are a plethora of Rootkits in existence for Unix platforms that contain all this functionality, because the NT rootkit is sti ll under development, its functionality is yet limited. That does not mean to say it is benign however.

What Is the NT Rootkit Capable Of?

The following readme.txt came with the 0.40 version used for this testing. It gives a good idea of the sort of nasty tricks that can be performed even with this early version of NT Root:

```
Alpha build - debug 0.40

This has been tested and known to work under NT 4.0 Server (1381).
This has been tested and known to work under Windows 2000 RC2 (2128).

Note: this debug buil d of the rootkit generates huge amounts of debug
messages.  You can watch these with a tool such as DbgView from
www.sysinternals.com (or equivalent).

To test out the rootkit, copy deploy.exe and _root_.sys to a common
directory.

To install and start the  rootkit, run deploy.exe.

To start and stop the rootkit in realtime, use the following commands:

net start _root_
net stop _root_

Respectively.

NEWS
----

Keyboard sniffing has been disabled for now.  You can comment the line back
in DriverEntry() if y our daring. Keyboard sniffing actually works fine   -
except that it has caused a BSOD on one of my test machines and I didn't
want to release it that way until the problem could be debugged.

New features:

Embedded TCP/IP stack (stateless)
---------------- -----------------

NT ROOTKIT has a stateless TCP/IP stack.  It works by determining the state
of the connection based on the data within the incoming packet.  This wokrs
fine for all tests we have performed on the local segment.  This has not
yet been tes ted over great distances of Internet.

The ROOTKIT has a hardcoded IP address to which it will respond.  As
delivered, this IP address is 10.0.0.166  - if you have a client machine
that is configured with a 10.X address, it should be able to telnet to the
rootkit.  Keep in mind that the rootkit is using raw connections to your
ethernet so it can do some amazing things.  First you will notice that the
target port does not matter.  You can telnet to any port and it will work.
Second - you will notice that mul tiple people can log into the rootkit at
once.  The sessions are not kept seperate but testing has shown that it
seems to work quite well as long as two people aren't typing commands at
exactly the same time.
```

```
NOTE: THIS MEANS THAT ROOTKIT DOES NOT SHOW UP   IN NETSTAT

Ideed, why would it?  It's not using the NT stack.

Gotcha: The rootkit IP address has better not conflict with a real machine
on your network, else the two will get into an ARP war    - and that is not
good.  Get this: the rootkit needs to use a   unique IP address~!


Command Shell
-------------

We have experimented with launching win32 processes from kernel mode.  This
has been non -trivial.  We have demonstrated this working at Blackhat    - but
the feature is disabled in this build.  It will be ad  ded back in for the
044 branch  - but there are many kinks still being worked out.

HIDE PROCESSES
--------------

Any process that starts with '_root_' will be hidden.  This feature can be
toggled on/off from the k -mode shell.  Just login and type 'hidepro  c' to
toggle.

HIDE FILES AND DIRS
-------------------

Any directory or file that starts with '_root_' will be hidden.  This
feature can be toggled from the k -mode shell.  Just login and type
'hidedir' to toggle.

Processes that are named with a prefix of   '_root_' are excempt from these
rules.  This means if your running a shell as '_root_cmd.exe' you can still
see the hidden stuff.  This means that '_root_taskmgr.exe' can still see
hidden processes.


Test EXE redirection:
---------------------

For now, this test is hard coded.  To test, first carry out the following:

Copy 'calc.exe' to C: \
Copy any other executable to C: \ and rename it so that the first 6
characters of the filename are '_root_'.  CMD.EXE was tested, so it would
be ranamed to "C: \_root_cmd.exe".

The rootkit will detect the execution of the filename that starts with
'_root_' and redirect it to "C: \calc.exe".  Try executing the file and you
will see that calc.exe gets executed instead.

Now, with the rootkit turned off, open '_root_cmd.exe  ' (or equivalent) in a
hex editor.  Now start the rootkit and open it again.  Note that the images
are exactly the same!  You are looking at the same file.  Now open calc.exe
and verify that it is different.  As you can see the rootkit does not
effect the  ability to read a file correctly.  The rootkit only becomes
involved when the file is executed.  This should fool programs that perform
CRC's or Hashes of files.


Test Registry Hiding:
```

```
---------------------

Any value or key that begins with the 6 lette  rs '_root_' should be hidden
from view. regedit.exe and regedt32.exe were tested.

Additionally, any program that is running that begins with '_root_' will be
excempt from any subterfuge  - hence, if you make a copy of regedit.exe
called '_root_regedit.exe'  - the new copy of regedit will be able to see
all of the hidden keys!  (neato)

Try starting and stopping the rootkit dynamically and refreshing your view
of the registry, also.  You will see that it is working.
```

When a host is compromised, the rootkit run s in kernel mode with system privileges and consequently has access to all resources the operating has access to. This makes it possible to:

- Hide files
- Hide processes
- Hide registry entries
- Intercept keystrokes
- Redirect *.exe* files

The following screen shots demonstrate how the NT rootkit can hide files from Windows Explorer and the *dir* command:

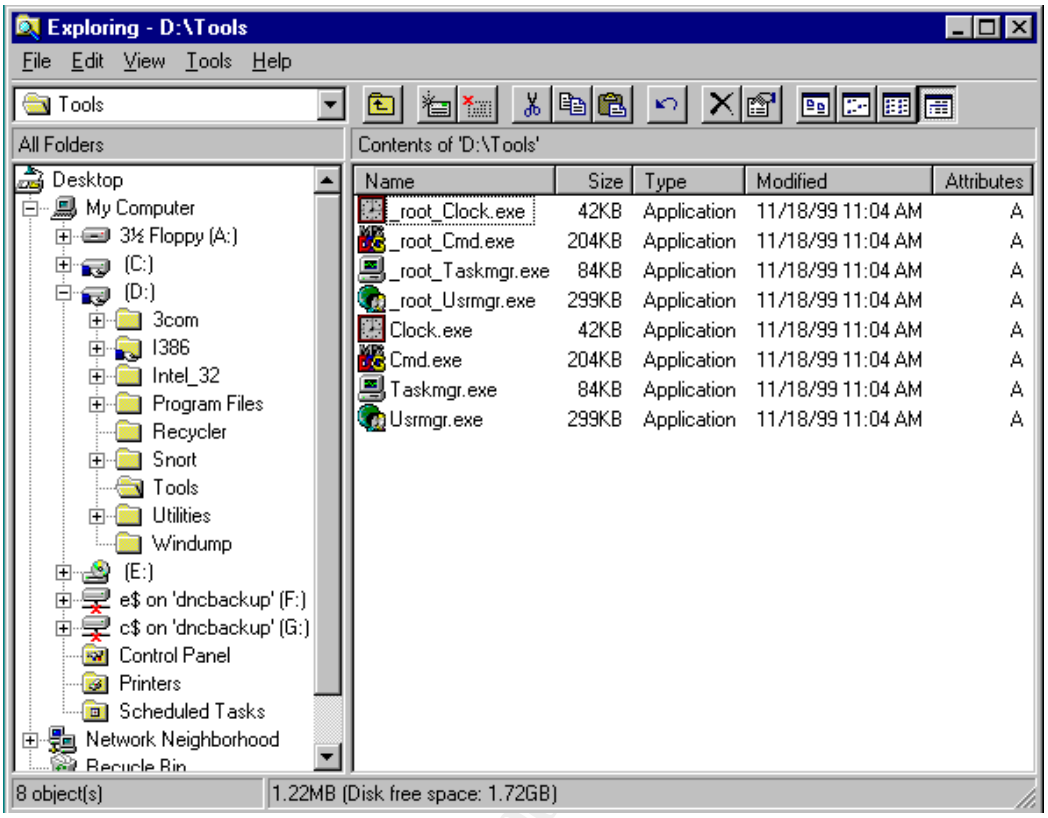

Figure 1: Windows Explorer's view without NT rootkit running

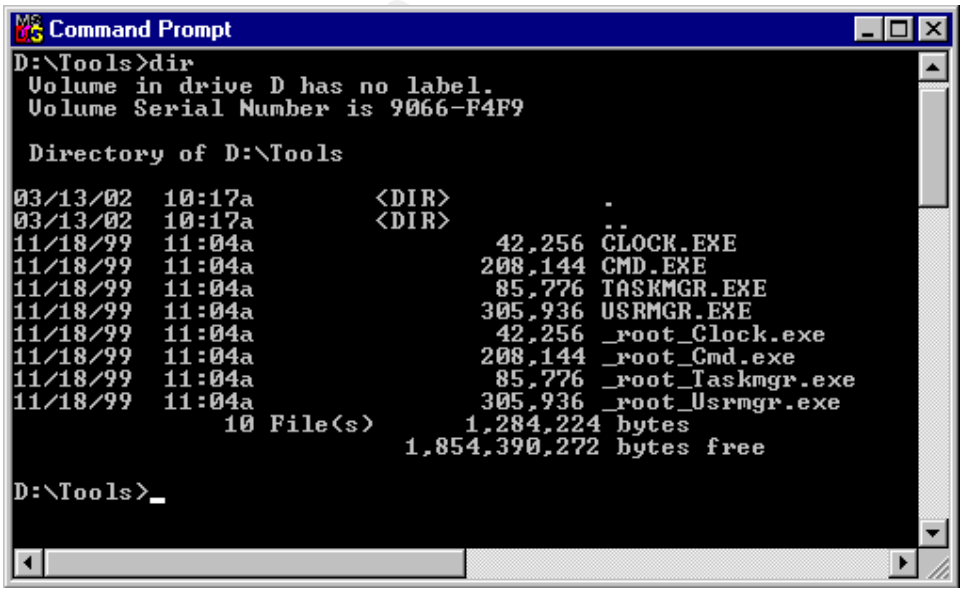

Figure 2: *dir* listing without NT rootkit running

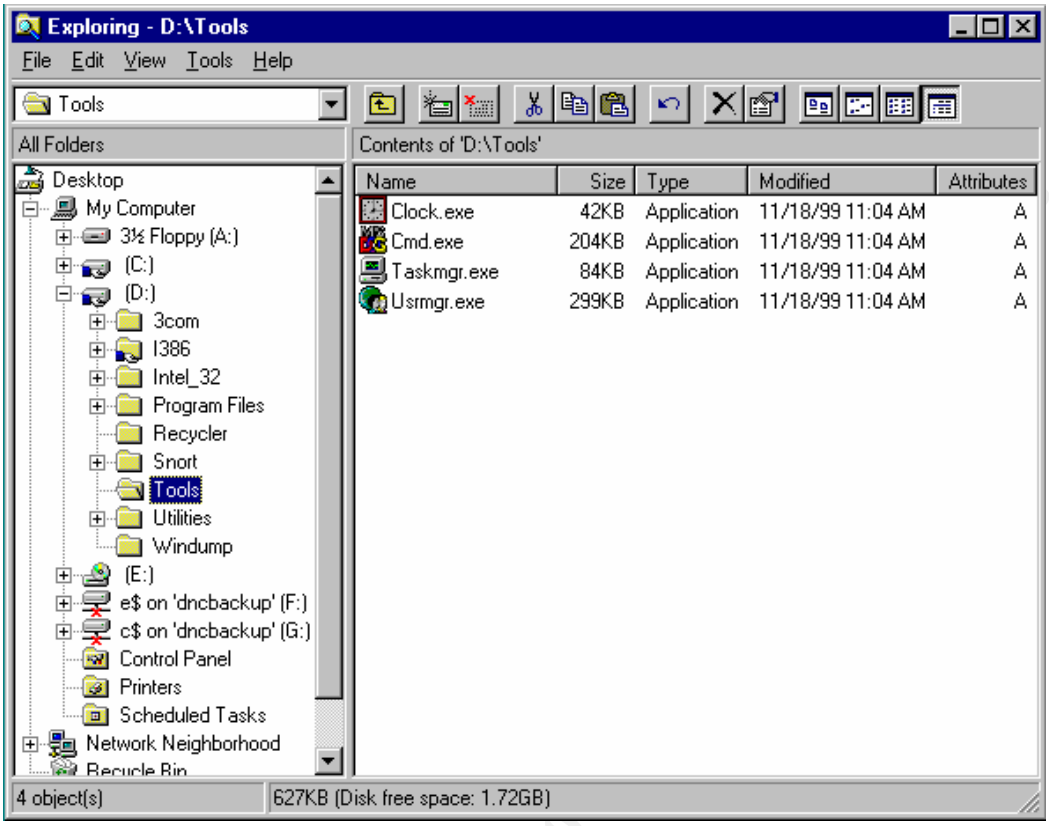

Figure 3: Windows Explorer's view with NT rootkit running

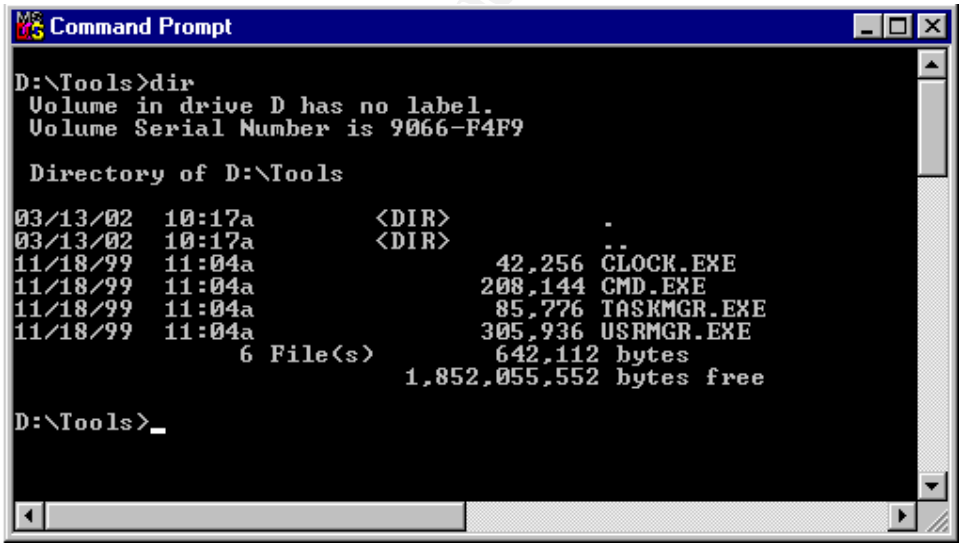

Figure 4: *dir* listing with NT rootkit running

The rootkit also assigns the host an additional IP address (for vers ion 0.40 this is hard coded to 10.0.0.166) and even assigns an additional MAC address (for version 0.40 this is hard coded to the rather novel DE:AD:BE:EF:DE:AD). This IP address

does not respond to ICMP echo requests and will not be detected by most network scans, so is essentially invisible on the network.

The attacker can telnet to this IP address on any port and execute any of the trojanised commands with little chance of being detected.

The NT rootkit can even sniff keystrokes, including the Ctrl-Alt-Del sequence. This makes it possible for an attacker to obtain passwords (including Administrator) in clear text as they are typed in.

This gives an idea of the power of the rootkit, even at this stage of development. As hackers become more familiar with the Windows operating systems by reverse engineering, they will become more equipped to find and utilise vulnerabilities with future rootkits.

For much more detailed further reading, see Greg Hoglund's article in Phrack magazine (quoted above) for a detailed explanation of how the NT Rootkit works.

How Could My Windows System Become Compromised?

For an attacker to install the NT rootkit on your computer they need to gain administrative access. Once this access is gained, all that is needed is to copy the following two files onto the computer:

- `_root_.sys`
- `deploy.exe`

The attacker then executes the command:

```
deploy.exe.
```

Once the rootkit has been installed `deploy.exe` can be deleted to cover his tracks.

The rootkit can then be started using:

- `Net start _root_`

And stopped by:

- `Net stop _root_`

Please note: the methods a hacker may use to obtain administrative access are beyond the scope of this document.

How Do I Know If My System Has Been Compromised?

Rootkits are designed to go unnoticed. Consequently, they are not always easy to detect unless the Administrator knows what to look for and where to look, or has the appropriate tools available.

There are a number of ways to determine if a system is compromised. Some suggestions are provided on the *CNET Builder.com* Website.[5]

This paper will consider how the presence of the NT rootkit can be detected from a network traffic perspective. Below is a dump from Snort of traffic taken from a Telnet session controlling a compromised computer across a LAN.

First, we see the initial 3-way handshake connection sequence between the attacker's workstation and the compromised host. While Snort is not capable of alerting on MAC addresses with the add-ons the author is aware of, a manual check of the logs would reveal the suspicious host MAC address of DE:AD:BE:EF:DE:AD. To be certain, a search on the Internet revealed that the DE:AD:BE range is not assigned to any vendor. Otherwise, there is nothing here that would raise any alarms

(Please note, to avoid the following traces wrapping, the r eader could reduce the font size to 9 points):

```
02/15-16:45:02.541999 0:C0:4F:87:D9:DA  -> DE:AD:BE:EF:DE:AD
type:0x800 len:0x3C
10.0.0.1:1985  -> 10.0.0.166:23  TCP TTL:128 TOS:0x0 ID:21782
IpLen:20 DgmLen:44 DF
******S* Seq: 0x9071D  Ack: 0x0  Win: 0x2000   TcpLen: 24
TCP Options (1) => MSS: 1460

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:02.542649 DE:AD:BE:EF:DE:AD  -> 0:C0:4F:87:D9:DA
type:0x800 len:0x3C
10.0.0.166:23  -> 10.0.0.1:1985  TCP TTL:128 TOS:0x0 ID:2178 2
IpLen:20 DgmLen:44 DF
***A**S* Seq: 0x0  Ack: 0x9071E  Win: 0x4470  TcpLen: 24
TCP Options (1) => MSS: 1460

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:02.542659 DE:AD:BE:EF:DE:AD  -> 0:C0:4F:87:D9:DA
type:0x800 len:0x3C
10.0.0.166:23  -> 10.0.0.1:1985  TCP TTL:128 TOS:0x0 ID:21782
IpLen:20 DgmLen:45 DF
***AP*** Seq: 0x1  Ack: 0x9071E  Win: 0x4470  TcpLen: 24
TCP Options (1) => MSS: 1460
A8                                                  ¨

=+=+=+=+=+=+=+=+=+=+=+= +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+
```

Now come the instructions sent via Telnet, one character at a time, echoed back to the attacker's workstation also. If we reassembled the packets we could piece together the instructions being sent via t he payload to the compromised host and being echoed back to the source ("hidedir"):

```
02/15-16:45:03.869604 0:C0:4F:87:D9:DA  -> DE:AD:BE:EF:DE:AD
type:0x800 len:0x3C
10.0.0.1:1985  -> 10.0.0.166:23  TCP TTL:128 TOS:0x0 ID:22550
IpLen:20 DgmLen:41 DF
***AP*** Seq: 0x9071E  Ack: 0x2  Win: 0x2237  TcpLen: 20
68                                                    h

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+
```

```
02/15-16:45:03.870084 DE:AD:BE:EF:DE:AD  -> 0:C0:4F:87:D9:DA
type:0x800 len:0x3C
10.0.0.166:23  -> 10.0.0.1:1985 TCP TTL:128 TOS:0x0 ID:22550
IpLen:20 DgmLen:41 DF
***AP*** Seq: 0x2  Ack: 0x9071F  Win: 0x2236  TcpLen: 20
68                                                    h

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:04.037644 0:C0:4F:87:D9:DA  -> DE:AD:BE:EF:DE:AD
type:0x800 len:0x3C
10.0.0.1:1985  -> 10.0.0.166:23 TCP TTL:128 TOS:0x0 ID:22806
IpLen:20 DgmLen:41 DF
***AP*** Seq: 0x9071F  Ack: 0x3  Win: 0x2236  TcpLen: 20
69                                                    i

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:04.038097 DE:AD:BE:EF:DE:AD  -> 0:C0:4F:87:D9:DA
type:0x800 len:0x3C
10.0.0.166:23  -> 10.0.0.1:1985 TCP TTL:128 TOS:0x0 ID:22806
IpLen:20 DgmLen:41 DF
***AP*** Seq: 0x3  Ack: 0x90720  Win: 0x2235  TcpLen: 20
69                                                    i

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:04.168663 0:C0:4F:87:D9:DA  -> DE:AD:BE:EF:DE:AD
type:0x800 len:0x3C
10.0.0.1:1985  -> 10.0.0.166:23 TCP TTL:128 TOS:0x0 ID:23062
IpLen:20 DgmLen:40 DF
***A**** Seq: 0x90720  Ack: 0x4  Win: 0x2235  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:04.205738 0 :C0:4F:87:D9:DA -> DE:AD:BE:EF:DE:AD
type:0x800 len:0x3C
10.0.0.1:1985  -> 10.0.0.166:23 TCP TTL:128 TOS:0x0 ID:23318
IpLen:20 DgmLen:41 DF
***AP*** Seq: 0x90720  Ack: 0x4  Win: 0x2235  TcpLen: 20
64                                                    d

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:04.206188 DE:AD:BE:EF:DE:AD  -> 0:C0:4F:87:D9:DA
type:0x800 len:0x3C
```

```
10.0.0.166:23 -> 10.0.0.1:1985 TCP TTL:128 TOS:0x0 ID:23318
IpLen:20 DgmLen:41 DF
***AP*** Seq: 0x4  Ack: 0x 90721  Win: 0x2234  TcpLen: 20
64                                               d

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:04.368964 0:C0:4F:87:D9:DA  -> DE:AD:BE:EF:DE:AD
type:0x800 len:0x3C
10.0.0.1:1985 -> 10.0.0.166:23 TCP TTL:128 TOS:0x0 ID:23574
IpLen:20 DgmLen:40 DF
***A**** Seq: 0x90721  Ack: 0x5  Win: 0x2234  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:04.423321 0:C0:4F:87:D9:DA  -> DE:AD:BE:EF:DE: AD
type:0x800 len:0x3C
10.0.0.1:1985 -> 10.0.0.166:23 TCP TTL:128 TOS:0x0 ID:23830
IpLen:20 DgmLen:41 DF
***AP*** Seq: 0x90721  Ack: 0x5  Win: 0x2234  TcpLen: 20
65                                               e

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ =+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:04.423806 DE:AD:BE:EF:DE:AD  -> 0:C0:4F:87:D9:DA
type:0x800 len:0x3C
10.0.0.166:23 -> 10.0.0.1:1985 TCP TTL:128 TOS:0x0 ID:23830
IpLen:20 DgmLen:41 DF
***AP*** Seq: 0x5  Ack: 0x90722  Win: 0x2233  TcpLen: 20
65                                               e

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:04.569240 0:C0:4F:87:D9:DA  -> DE:AD:BE:EF:DE:AD
type:0x800 len:0x3C
10.0.0.1:1985 -> 10.0.0.166:23 TCP TTL:128 TOS:0x0  ID:24086
IpLen:20 DgmLen:40 DF
***A**** Seq: 0x90722  Ack: 0x6  Win: 0x2233  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:05.080489 0:C0:4F:87:D9:DA  -> DE:AD:BE:EF:DE:AD
type:0x800 len:0x3C
10.0.0.1:1985 -> 10.0.0.166:23 TCP TTL:128 TOS:0x0 ID:24342
IpLen:20 DgmLen:41 DF
***AP*** Seq: 0x90722  Ack: 0x6  Win: 0x2233  TcpLen: 20
```

```
64                                                          d

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:05.080953 DE:AD:BE:EF:DE:AD  -> 0:C0:4F:87:D9:DA
type:0x800 len:0x3C
10.0.0.166:23  -> 10.0.0.1:1985 TCP TTL:128 TOS:0x0 ID:24342
IpLen:20 DgmLen:41 DF
***AP*** Seq: 0x6  Ack: 0x90723  Win: 0x2232  TcpLen: 20
64                                                          d

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:05.261534 0:C0:4F:87:D9:DA  -> DE:AD:BE:EF:DE:AD
type:0x800 len:0x3C
10.0.0.1:1985  -> 10.0.0.166:23 TCP TTL:128 TOS:0x0 ID:24598
IpLen:20 DgmLen:41 DF
***AP*** Seq: 0x90723  Ack: 0x7  Win: 0x2232  TcpLen: 20
69                                                          i

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:05.261987 DE:AD:BE:EF:DE:AD  -> 0:C0:4F:87:D9:DA
type:0x800 len:0x3C
10.0.0.166:23  -> 10.0.0.1:1985 TCP TTL:128 TOS:0x0 ID:24598
IpLen:20 DgmLen:41 DF
***AP*** Seq: 0x7  Ack: 0x90724  Win: 0x2231  TcpLen: 20
69                                                          i

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:05.370406 0:C0:4F:87:D9:DA  -> DE:AD:BE:EF:DE:AD
type:0x800 len:0x3C
10.0.0.1:1985  -> 10.0.0.166:23 TCP TTL:128 TOS:0x0 ID:24854
IpLen:20 DgmLen:40 DF
***A**** Seq: 0x90724  Ack: 0x8  Win: 0x2231  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:05.406095 0:C0:4F:87:D9:DA  -> DE:AD:BE:EF:DE:AD
type:0x800 len:0x3C
10.0.0.1:1985  -> 10.0.0.166:23 TCP TTL:128 TOS:0x0 ID:25110
IpLen:20 DgmLen:41 DF
***AP*** Seq: 0x90724  Ack: 0x8  Win:  0x2231  TcpLen: 20
72                                                          r
```

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:05.406602 DE:AD:BE:EF:DE:AD -> 0:C0:4F:87:D9:DA
type:0x800 len:0x3C
10.0.0.166:23 -> 10.0.0.1:1985 TCP TTL:128 TOS:0x0 ID:25110
IpLen:20 DgmLen:41 DF
***AP*** Seq: 0x8  Ack: 0x90725  Win: 0x2230  TcpLen: 20
72                                                    r

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:05.570684 0:C0:4F:87:D9:DA -> DE:AD:BE:EF:DE:AD
type:0x800 len:0x3C
10.0.0.1:1985 -> 10.0.0.166:23 TCP TTL:128 TOS:0x0 ID:25366
IpLen:20 DgmLen:40 DF
***A**** Seq: 0x90725  Ack: 0x9  Win: 0x2230  TcpLen: 20
```

Finally comes the response from the compromised host. This time the payload is the confirmation the instruction was successful.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+

02/15-16:45:05.647805 DE:AD:BE:EF:DE:AD -> 0:C0:4F:87:D9:DA
type:0x800 len:0x57
10.0.0.166:23 -> 10.0.0.1:1985 TCP TTL:128 TOS:0x0 ID:25622
IpLen:20 DgmLen:73 DF
***AP*** Seq: 0xB  Ack: 0x90727  Win: 0x222E  TcpLen: 20
64 69 72 65 63 74 6F 72 79 20 70 72 65 66 69 78  directory
prefix
2D 68 69 64 69 6E 67 20 6E 6F 77 20 4F 46 46 0D   -hiding now
OFF.
0A                                                    .
```

Aside from the payload and the MAC address, there is nothing suspicious about this traffic flow, making detection difficult. The only sure way of detecting an attacker accessing a system compromised with NT Root v ersion 0.40 would be to check the payload of the TCP packets.

What Do I Do If My System Is Compromised?

A variety of tools such as *The Cleaner* from *Moosoft*[6] claim to be able to detect and clean rootkits from your system just as anti -virus software can clea n a computer infected by a virus. However, unless a Trojan Cleaner comes from a known trusted source, there is no guarantee it does not contain malicious code itself, so users must be very wary.

For the current development version of the NT rootkit, start by ensuring the `_root_` service is not running by typing the following at a command prompt:

```
Net stop _root_
```

Next search for and remove the following:

Files and directories starting with _root_, e.g. _root_cmd.exe, _root_regedit.exe, or _root_taskmgr.exe.

Registry entries starting with `_root_`.

Processes starting with `_root_`, using Task Manager.

How Can I Protect Against NT Rootkits?

The following steps will provide protection from NT Root attacks:

1. Block all unnecessary ports at your perimeter and allow specific ports access only to the IP addresses of those servers that require them. E.g. only allow port 25 access to mail servers, etc. [7]
2. Use a proxy server for Internet access. This way you can limit which hosts can have access to the Internet and ensure that rogue IP addresses (like the one created by NT Root) aren't accessible from outside your network.
3. Block traffic entering your network destined for subnets that don't exist on your network.
4. Run reliable anti-virus software on all of your hosts and keep it up to date. Most anti-virus software will detect known rootkits entering your network. (I had to turn off my anti-virus software to download the rootkit).
5. Run an integrity-checking intrusion detection tool, such as *TripWire* to produce a baseline of your systems (remember to ensure you make the baseline from a clean system – if you are in any doubt about the integrity of your system, you would be wise to reinstall it before proceeding). These tools need to be complemented by a trusted source of critical files. This could be either a regular backup (make sure the one you restore from was taken before the last good checksum was run to ensure its integrity). Alternatively, you could save critical files on a read only medium, such as CD-ROM.

   Since it is even possible for future Rootkits to compromise integrity checking software, the following advice from CNET would be worth adopting:

   "Your library of safe MD5 checksums should be calculated from sources that the attacker cannot also modify, either original CD-ROMs or checksums direct from vendors via the Web. Also --and this is crucial --when calculating the checksum of suspected Trojan horse binaries, do not *use a copy of the MD5 program that is on the victim system. What if the crafty attacker also sabotaged the MD5 program to display incorrect checksums for given binaries? You should use a known good copy of MD5 from media such as CD-ROM or floppy diskette instead. In fact, as part of normal system administration duties, it is a great idea to save a known good copy of all important system utilities to a CD-ROM or floppy. This way, in the event of an incident, you can use the good saved copies of the binaries to investigate. You can also automate the process of creating and comparing checksums with Tripwire."* [8]
6. Use an IDS like Snort with a custom rule set to check for the response strings as part of the payload. The following Snort rule will detect these events:

```
alert tcp any any -> any any (content: "Win2K Rootkit"; msg: "Possible NT
      Rootkit";)
alert tcp any any -> any any (content: "directory prefix-hiding"; msg:
      "Possible NT Rootkit";)
```

Things to Watch Out For in the Future

Because of a rootkit's ability to compromise the TCP/IP stack, a very likely way of sending instructions to compromised hosts would be via a covert channel. This could be implemented using crafted packets, possibly ICMP packets containing a payload, or through the use of unassigned fields in any IP packet. Checking for ICMP packets with a payload, or checking f or use of unassigned fields in different types of IP packets could potentially detect these compromises. Better still, only allow ICMP echo replies into your network, and not echo requests. Future instructions may even be encrypted, making it yet harder to detect.

References

[1] Windows NT rootkit: http://www.megasecurity.org/Tools/Nt_rootkit_all.html

[2] Snort: http://www.snort.org/downloads.html

[3] Rootkit definition: http://whatis.techtarget.com/definition/0,289893,sid9_gci547279,00.html

[4] Rootkit definition: http://www.phrack.org/show.php?p=55&a=5

[5] How to find NT rootkit: http://builder.cnet.com/webbuilding/0-7532-8-4996985-1.html?tag=st.bl.7532.edt.7532-8-4996985-1

[6] *The Cleaner* by *Moosoft*: http://www.moosoft.com/download.php

[7] Perimeter security: http://rr.sans.org/firewall/blocking_cisco.php

[8] Detecting Rootkits: http://builder.cnet.com/webbuilding/0-7532-8-4720241-2.html?tag=st.bl.7532-8-4720241-1.txt.7532-8-4720241-2

Assignment 2 – Network Detects

Detect 1 – Scan for Web Servers

The following extracts show the beginning and ending of scan activity was detected on my network. The number following each set is the number of probes for that source. T imestamps are GMT-0600.

```
Feb 14 04:12:26 217.136.114.162:3408  -> xxx.yyy.0.0:80 SYN
******S*
Feb 14 04:12:29 217.136.114.162:3409  -> xxx.yyy.0.1:80 SYN
******S*
Feb 14 04:12:26 217.136.114.162:3410  -> xxx.yyy.0.2:80 SYN
******S*
Feb 14 04:12:26 217.136.114. 162:3416 -> xxx.yyy.0.8:80 SYN
******S*
Feb 14 04:12:29 217.136.114.162:3417  -> xxx.yyy.0.9:80 SYN
******S*
Feb 14 04:12:26 217.136.114.162:3419  -> xxx.yyy.0.11:80 SYN
******S*
Feb 14 04:12:29 217.136.114.162:3420  -> xxx.yyy.0.12:80 SYN
******S*
Feb 14 04:12:29 217.136.114.162:3422  -> xxx.yyy.0.14:80 SYN
******S*
[...]
Feb 14 05:14:06 217.136.114.162:4404  -> xxx.yyy.67.100:80 SYN
******S*
Feb 14 05:14:10 217.136.114.162:4413  -> xxx.yyy.67.173:80 SYN
******S*
Feb 14 05:14:12 217.136.114.162:4419  -> xxx.yyy.67.173:80 SYN
******S*
Feb 14 05:14:17 217.136.114.162:4431  -> xxx.yyy.67.173:80 SYN
******S*
Feb 14 05:14:21 217.136.114.162:4442  -> xxx.yyy.67.173:80 SYN
******S*
Feb 14 05:14:22 217.136.114.162:4444  -> xxx.yyy.67.173:80 SYN
******S*
Feb 14 05:14:23 217.1 36.114.162:4446 -> xxx.yyy.67.184:80 SYN
******S*
Feb 14 05:14:25 217.136.114.162:4448  -> xxx.yyy.71.250:80 SYN
******S*
42975
```

Source of Trace

This trace was obtained from http://www.incidents.org/archives/intrusions/msg03828.html and was posted by Ken Connelly on Fri, 15 Feb 2002

Detect Was Generated By

The detect appears to be generated by Snort running with the *–A fast* option set to minimise the output. Following is an explanation of the Snort logs in this mode:

```
Date, Time, First.IP.Address:Port Number, ->(direction of data
flow), Second.IP.Address:Port Number, Flags
```

Probability the Source Address Was Spoofed

The probability the source IP address was spoofed in this scan is very low. If the attacker had spoofed the address, he would not receive any response indicating a Web Server was found. It is possible the response coud be sent to another network the attacker is monitoring. But, considering the level of skill required to compromise many Web Servers, this attacker is unlikely to have such extensive resources at his disposal.

Description of Attack

The attacker is apparrently scanning for Web servers, looking for any hosts that respond to stimulation on port 80. If the attacker does find any Web Servers his next step would be to determine what type of Web Servers are found and then attempt the respective exploits. It is most likely the attacker would be targetting a particular type of Web server, such as Microsoft IIS, using exploits he is familiar with.

Attack Mechanism

Either a TCP connect scan or a TCP SYN scan is being performed. A TCP connect scan is the most basic scan and is an attempt to establish a three-way handshake with the destination host on the chosen port. A TCP SYN scan is a litlle more stealthy. After the destination responds with a SYN and ACK, the source does not complete the three-way handshake with an ACK, leaving the connection half open. Some firewall and intrusion detection systems will not detect this sort of scan. Because this type of scan has become so common, most systems can detect it now.

Since the scan appears to be systematically covering an entire class B address space, it has probably been performed with a port scanning tool, similar to Nmap or ScanPort. It is difficult to determine exactly which tool was used without more information, such as TCP sequence and ACK numbers and IP ID.

For the initial part of the scan, the source port increments at the same rate as the destination IP address. This another likely indicator of a port scanning too in use.

Correlations

Dshield reports 5 instances of IP addresses from the 217.136.114.0 subnet targetting port 80. No other specific reports could be found of this IP address targetting port 80.

Evidence of Active Targetting

This attack actively scanned the range of addresses from xxx.yyy.0.0. to xxx.yyy.71.250. It is likely that the scan was wider than this also. The attacker was actively targetting port 80.

Severity

Target criticality: 4

The attacker is speficall y targetting Web servers.

Attack Lethality: 4

Due to the number of vulnerabilities associated with various Web Servers, this attack has the potential to gain root access on compromised hosts.

System Countermeasures: 3

Details of the targetted hosts is u nknown. This could range from 1 for unpatched Web servers of the type the attacker is targetting to 5 for patched Web servers that are not being targetted.

Network Countermeasures: 1

Details of the perimeter security are unknown. Assuming the IDS is inside the perimeter firewall the attack has reached the internal LAN.

Attack Severity: (4 + 4) – (3 + 1) = 4

Defensive recommendation

Ensure external access to port 80 is allowed only to Web Servers. Ensure all Web Servers are patched against all known vulnerab ilities.

Multiple Choice Test Question

What is the following trace evidence of?

```
Feb 14 04:12:26 217.136.114.162:3408  -> xxx.yyy.0.0:80 SYN
******S*
Feb 14 04:12:29 217.136.114.162:3409  -> xxx.yyy.0.1:80 SYN
******S*
Feb 14 04:12:26 217.136.114.162:3410  -> xxx.yyy.0.2:80 SYN
******S*
 [...]
Feb 14 05:14:22 217.136.114.162:4444  -> xxx.yyy.67.173:80 SYN
******S*
Feb 14 05:14:23 217.136.114.162:4446  -> xxx.yyy.67.184:80 SYN
******S*
Feb 14 05:14:25 217.136.114.162:4448  -> xxx.yyy.71.250:80 SYN
******S*
```

A. The attacker is attempting to exploit a buffer overflow vulnerability on a Microsoft IIS Web Server.
B. The source address is probably spoofed.
C. The attacker is scanning for Web Servers.
D. The TCP flag settings are out of spec.

Detect 2 – Potential IRC Trojan Activity

Hi,

I have been checking our syslogs daily and have seen the following entries daily. I have done some checks on the web for ports 4400 and the other ports, to no avail. Not sure what to make of it.. Any ideas?

```
Feb 14 12:46:38,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15912
Feb 14 12:47:10,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15942
Feb 14 12:47:38,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/80 dst
outside:aa.bb.226.dd/15890
Feb 14 12:47:42,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15912
Feb 14 12:48:02,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15929
Feb 14 12:48:10,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15943
Feb 14 12:48:14,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15942
Feb 14 12:48:42,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp sr> outside:168.167.25.2/80 dst outside:aa.bb.226.dd/15890
Feb 14 12:48:46,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15912
Feb 14 12:49:06,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15929
Feb 14 12:49:14,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15943
Feb 14 12:49:18,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15942
Feb 14 12:49:46,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/80 dst
outside:aa.bb.226.dd/15890
```

```
Feb 14 12:49:50,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15912
Feb 14 12:49:50,  , 3, %PIX -3-106011: Deny inbound  (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15912
Feb 14 12:50:10,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15929
Feb 14 12:50:18,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15943
Feb 14 12:50:22,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15942
Feb 14 12:50:50,  , 3, %PIX -3-106011: Deny inbound  (No xlate)
tcp src> outside:168.167.25.2/80 dst
outside:aa.bb.226.dd/15890
Feb 14 12:50:50,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/80 dst
outside:aa.bb.226.dd/15890
Feb 14 12:50:54,  , 3, %PIX -3-106011: Deny inbound (No  xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15912
Feb 14 12:51:14,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15929
Feb 14 12:51:14,  , 3, %PIX -3-106011: Deny inbound (No  xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15929
Feb 14 12:51:22,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15943
Feb 14 12:51:26,  , 3, %PIX -3-106011: Deny inbound (No  xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15942
Feb 14 12:51:54,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/80 dst
outside:aa.bb.226.dd/15890
Feb 14 12:51:58,  , 3, %PIX -3-106011: Deny inbound (No x late)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15912
Feb 14 12:52:18,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15929
```

```
Feb 14 12:52:26,  , 3, %PIX -3-106011: Deny inbound (No x late)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15943
Feb 14 12:52:30,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15942
Feb 14 12:52:36,  , 3, %PIX -3-106011: Deny inbound (No x late)
tcp src> outside:168.167.25.2/80 dst
outside:aa.bb.226.dd/15890
Feb 14 12:53:02,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15912
Feb 14 12:53:10,  , 3, %PIX -3-106011: Deny inbound (No xla te)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15929
Feb 14 12:53:30,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15943
Feb 14 12:53:34,  , 3, %PIX -3-106011: Deny inbound (No xla te)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15942
Feb 14 12:53:46,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/80 dst
outside:aa.bb.226.dd/15890
Feb 14 12:54:06,  , 3, %PIX -3-106011: Deny inbound (No xlate )
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15912
Feb 14 12:54:26,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15929
Feb 14 12:54:34,  , 3, %PIX -3-106011: Deny inbound (No xlate )
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15943
Feb 14 12:54:38,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15942
Feb 14 12:54:46,  , 3, %PIX -3-106011: Deny inbound (No xlate )
tcp src> outside:168.167.25.2/80 dst
outside:aa.bb.226.dd/15890
Feb 14 12:55:10,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15912
Feb 14 12:55:30,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15929
```

```
Feb 14 12:55:38,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15943
Feb 14 12:55:42,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15942
Feb 14 12:56:34,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15929
Feb 14 12:56:42,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15943
Feb 14 12:56:46,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15942


Simon Roper
```

Source of Trace

This trace was obtained from http://www.incidents.org/archives/intrusions/msg03824.html and was posted by Simon Roper on Friday, February 15, 2002

Detect Was Generated By

Cisco PIX firewall (version unknown). Fol lowing is an explanation of the Cisco PIX logs:

```
Timestamp, firewall name, PIX message severity and number:
firewall action, source  interface:ip/port, destination
interface:ip/port
```

Probability the Source Address Was Spoofed

In either of the two possibilitie s described below, the probability of the source address being spoofed is low. Either an IRC server is communicating with a host compromised with a trojan, which would require a three -way handshake, or it is normal IRC traffic.

Description of Attack

%PIX-7-106011: Deny inbound (No xlate) *chars*

Explanation *This is a connection -related message. This message occurs when a packet is sent to the same interface that it arrived on. This usually indicates that a security breach is occurring. When the PIX Firewall receives a packet, it tries to establish a translation slot based on the security policy you set with the* ***global*** *and* ***conduit*** *commands, and your routing policy set with the* ***route*** *command.*

Failing both policies, PIX Firewall allows the packet to flow from the higher priority network to a lower priority network, if it is consistent with the security policy. If a packet comes from a lower priority network and the security policy does not allow it, PIX Firewall routes the packet back to the same interface.

*To provide access from an interface with a higher security to a lower security, use the **nat** and **global** commands. For example, use the **nat** command to let inside users access outside servers, to let inside users access perimeter servers, and to let perimeter users access outside servers.*

*To provide access from an interface with a lower security to higher security, use the **static** and **conduit** commands. For example, use the **static** and **conduit** commands to let outside users access inside servers, outside users access perimeter servers, or perimeter servers access inside servers.*

***Action** Fix your configuration to reflect your security policy for handling these attack events.*

Performing a seach with Google revealed that port 4400 is used by several IRC servers, particularly related to online games, such as "Half-Llife" and "XPilot". This port is apparently being used as an alternative to the normal 6660-6669 range. For more information on this please refer to the following Web sites:

- Xpilot: http://bau2.uibk.ac.at/erwin/NM/www/source/metaserver.html
- Use of port 4400 rather than the tradtional 6600 – 6669 range: http://www.geocities.com/athens/3615/iirc.html (You will need to search for "4400" on this page.)

A number of trojans are asociated with IRC clients, so potentially this traffic could be response to a compromised host.

Alternatively, this traffic could simply be a response to connections established from within the network to an external IRC server located in Botswana.

The Cisco Web site provides the following description of a similar error, the only difference being the one in the trace has an error rating of 3 rather than 7:

Attack Mechanism

If this activity is the result of an IRC trojan, the trojan could have been contained within the IRC client installed on the host, or the user may have inadvertantly installed it from an attachment he received through the IRC contacts.

The following generic description of this type of trojan is given at http://www.hackfix.org/ircfix :

These trojans are different than normal trojans in the way that someone else controls your infected computer.

With most other trojans, they open a port on your system that a hacker needs to connect to (and thus know your systems internet host, or IP.)

IRC related trojans however, will open a hidden connection from your PC to an IRC server, where it will tell the hacker, or a group of hackers (or possibly even a very large channel of people) what your infected with, what your IP is, and any other information they program it to give.

Then these users can send commands to the hidden IRC connection, and tell your computer to do things, similar to other trojans.

These IRC trojans can range anywhere from so simple, that the users on IRC can only control that IRC connection (Usually using it to harass and abuse other users on IRC.) all the way to being able to run other programs on your computer, and installing other types of trojans.

Correlations

Dshield lists 4 records against this IP, but none relating to port 4400. No other correlations were found.

Evidence of Active Targetting

If this host has been compromised with a trojan, the evidence would suggest the host was known to be compromised and is being actively targetted by the attacker. Alternatively, it would simply be a response to attempt to connect to an IRC server.

Severity

Target criticailty: 1

Probably a respons e to a workstation, not a server.

Attack Lethality: 4

Considering the risk of trojan infection associated with some IRC clients the lethality could be high, resulting in compromised hosts. Alternatively, the lethality depends on view of organisation on all owing staff to access IRC, especially online games.

System Countermeasures: 1

Assuming the attack is the result of trojan activity, it would appear as though the host has already been compromised. If it is simply IRC communication, the system countermeasu re may be higher depending on the organisation's view of IRC usage and any known vulnerabilities in the IRC client.

Network Countermeasures: 3

The PIX firewall is blocking the response traffic, however if it is trojan behaviour, the host has already been c ompromised. There are no logs of outbound traffic to the IRC server. The assumption, therefore, is that this traffic is not being detected or blocked at the perimiter.

Attack Severity: (1 + 4) – (1 + 3) = -1

Defensive Recommendation

Investigate the host aa .bb.226.dd for IRC software and run up -to-date anti-virus software against it to check for trojans. Decide whether to leave the IRC client on the host based upon the organisation's policies. Ensure all incoming and outgoing traffic is scanned for viruses. Install anti-virus software on all workstations and ensure it is updated regularily. Block access to external port 4400 if it is the organisation's policy to not allow IRC access, especially for online games.

Multiple Choice Test Question

What does the following trace demonstrate:

```
Feb 14 12:46:38,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15912
Feb 14 12:47:10,  , 3, %PIX -3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/4400 dst
outside:aa.bb.226.dd/15942
```

```
Feb 14 12:47:38,  , 3, %PIX-3-106011: Deny inbound (No xlate)
tcp src> outside:168.167.25.2/80 dst
outside:aa.bb.226.dd/15890
```

A. Possible trojan activity.
B. This detect shows a DoS attack against aa.bb.226.dd.
C. The attacker is scanning for IRC servers.
D. The internal host aa.bb.226.dd is acting as an IRC server.

Detect 3 – Code Red Worm

```
=-=-=-=-=-=-=-=-=-=-=-=

193.136.20.89 - - [12/Feb/2002:00:21:41 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNN NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u685
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
u9090%u9090%u8190%u00c3%u00 03%u8b00%u531b%u53ff%u0078%u0000%u0
0=a  HTTP/1.0" 400 331 " -" "-"


=-=-=-=-=-=-=-=-=-=-=-=

24.101.1.20 - - [12/Feb/2002:01:11:50 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN NNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u685
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
0=a  HTTP/1.0" 400 328


=-=-=-=-=-=-=-=-=-=-=-=

213.143.39.4 - - [12/Feb/2002:01:38:45 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u685
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
0=a  HTTP/1.0" 400 328


=-=-=-=-=-=-=-=-=-=-=-=

65.80.54.67 - - [12/Feb/2002:03:04:14 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNN NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u685
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
```

```
u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
0=a  HTTP/1.0" 400 329

=-=-=-=-=-=-=-=-=-=
203.198.89.250 - - [12/Feb/2002:04:31:22 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u 685
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
0=a  HTTP/1.0" 400 328

=-=-=-=-=-=-=-=-=-=
208.34.90.25 - - [12/Feb/2002:08:16:20 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNNNNN NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u685
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9 090%u6858%ucbd3%u7801%
u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
0=a  HTTP/1.0" 400 326
=-=-=-=-=-=-=-=-=-=
202.119.112.67 - - [12/Feb/2002:09:10:57 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNN NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u685
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
u9090%u9090%u8190%u0 0c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
0=a  HTTP/1.0" 400 329

=-=-=-=-=-=-=-=-=-=
200.23.236.62 - - [12/Feb/2002:11:42:20 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN NNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u685
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
0=a  HTTP/1.0" 400 326

=-=-=-=-=-=-=-=-=-=
194.126.46.12 - - [12/Feb/2002:15:21:01 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN NNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u685
```

```
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
0=a  HTTP/1.0" 400 326

=-=-=-=-=-=-=-=-=-=
61.129.76.43 - - [12/Feb/2002:15:54:17 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNN NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u685
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
0=a  HTTP/1.0" 400 325

=-=-=-=-=-=-=-=-=-=
24.237.255.11 - - [12/Feb/2002:18:00:12 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090 %u685
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
0=a  HTTP/1.0" 400 326

=-=-=-=-=-=-=-=-=-=
61.132.16.138 - - [12/Feb/2002:21:39:54 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNN NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u685
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801 %u9090%u6858%ucbd3%u7801%
u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
0=a  HTTP/1.0" 400 331 "-" "-"

=-=-=-=-=-=-=-=-=-=
211.167.64.137 - - [12/Feb/2002:23:34:17 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u685
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
0=a  HTTP/1.0" 400 328
```

Source of Trace

This is an extract from a trace submitted to http://www.incidents.org/archives/intrusion_s/msg03821.html by Laurie Zirkle on Thu, 14 Feb 2002.

Detect Was Generated By

PortSentry. Following is an explanation of these PortSentry logs:

```
Source IP, Date/Time, Payload
```

Probability the Source Address was Spoofed

There is a very low likelihood of a sp oofed source address in this case, since worms are designed to spread themselves after compromising a host.

Description of Attack

The payload of these packets are the signature of the Code Red worm. According to http://www.cert.org/advisories/CA -2001-19.html :

The "Code Red" worm is self -replicating malicious code that exploits a known vulnerability in Microsoft IIS servers…

The "Code Red" worm activity can be identified on a machine by the pre sence of the following string in a web server log files:

```
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNN NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNN%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801
%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8
b00%u531b%u53ff%u0078%u0000%u00=a
```

The presence of this string in a log file does not neccessarily indicate compr omise. Rather it only implies that a "Code Red" worm attempted to infect the machine.

Additionally, web pages on victim machines may be defaced with the following message:

```
HELLO! Welcome to http://www.worm.com! Hacked By Chinese!
```

The text of this page is stored exclusively in memory and is not written to disk. Therefore, searching for the text of this page in the file system may not detect compromise.

Attack Mechanism

The same Web site referenced under “Description of Attack” describes the attack mechanism as follows:

The "Code Red" worm attack proceeds as follows:

1. *The "Code Red" worm attempts to connect to TCP port 80 on a randomly chosen host assuming that a web server will be found. Upon a successful connection to port 80, the attacking host sends a craf ted HTTP GET request to the victim, attempting to exploit a buffer overflow in the Indexing Service described in CERT advisory CA-2001-13*
2. *The same exploit (HTTP GET request) is sent to each of the randomly chosen hosts due to the self-propagating nature of the worm. However, depending on the configuration of the host which receives this request, there are varied consequences.*
    - ***IIS 4.0 and 5.0 servers with Indexing service installed*** *will almost certainly be compromised by the "Code Red" worm.*

- ***Unpatched Cisco 600-series DSL routers*** *will process the HTTP request thereby triggering an unrelated vulnerability which causes the router to stop forwarding packets. [http://www.cisco.com/warp/public/707/cisco -code-red-worm-pub.shtml]*
- ***Systems not running IIS, but with an HTTP server listening on TCP port 80*** *will probably accept the HTTP request, return with an "HTTP 400 Bad Request" message, and potentially log this request in an access log.*

3. *If the exploit is successful, the worm begins executing on the victim host. In the earlier variant of the worm, victim hosts with a default language of English experienced the following defacement on all pages requested from the server:*
4. *HELLO! Welcome to http://www.worm.com! Hacked By Chinese!*

Servers configured with a language that is not English and those infected with the later variant will not experience any change in the served cont ent.

Other worm activity on a compromised machine is time senstive; different activity occurs based on the date (day of the month) of the system clock.

- *Day 1 - 19: The infected host will attempt to connect to TCP port 80 of randomly chosen IP addresses i n order to further propagate the worm.*
- *Day 20 - 27: A packet-flooding denial of service attack will be launched against a particular fixed IP address*
- *Day 28 - end of the month: The worm "sleeps"; no active connections or denial of service*

Correlations

Dshield reports of recent activity from some of the source subnets involved in this attack against port 80 are listed in table 1 below:

Source Subnet	No. of Targets	Period
24.101.1.0	2175	2002-02-22 / 2002-03-25
213.143.39.0	246	2002-03-05 / 2002-03-19
65.80.54.0	32	2002-02-23 / 2002-03-24
203.198.89.0	221	2002-03-08 / 2002-03-19
202.119.112.0	65	2002-02-24 / 2002-03-01
200.23.236.0	109	2002-02-26 / 2002-03-19
194.126.46.0	312	2002-03-08 / 2002-03-19
61.129.76.0	934	2002-03-03 / 2002-03-19
24.237.255.0	26	2002-03-06 / 2002-03-23
61.132.16.0	273	2002-02-23 / 2002-03-19
211.167.64.0	4003	2002-02-27 / 2002-03-19

Table 1: Subnets targetting port 80

While the above report does not definitely indicate Code Red activity, if the h osts at these addresses are infected, it is very likely. Code Red is a very widely spread worm and there are many reports of infection in the wild.

Evidence of Active Targetting

As reported above under "Attack Mechanism" the " *host attempts to connect to TC P port 80 of randomly chosen IP addresses in order to further propagate the worm* ". Consequently, this attack involves active targetting Microsoft IIS servers, but not specific networks or hosts.

Severity

Target criticailty: 4

The Code Red worm specifically targets Microsoft IIS servers.

Attack Lethality: 5

Some variants of the CodeRed (such as CodeRed II) leave backdoors which allow arbitrary execution of commands.

System Countermeasures: 5

Since this attack was detected by PortSentry, the hosts are running either Unix or Linux operating systems. Code Red only affects Microsoft IIS. The system countermeasrure may need to be increased if the site also contained Microsoft IIS servers. Then this would be graded depending on the level of patching on those server s.

Network Countermeasures: 1

Packets have reached the host before being detected.

Attack Severity: (4 + 5) – (5 + 1) = 3

Defense

If the site hosts any Microsoft IIS servers, ensure they are patched against all known vulnerabilities. Run anti-virus softwar e against all incoming and outgoing traffic and on internal hosts. Ensure the anit-virus software is updated on a regular basis.

Question

This attack targets the following:

```
193.136.20.89 - - [12/Feb/2002:00:21:41 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNNN NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u685
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u 9090%u6858%ucbd3%u7801%
u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
0=a  HTTP/1.0" 400 331 "-" "-"
```

A. Microsoft SQL Server 7.0 and 2000
B. Microsoft IIS 4.0 and 5.0
C. Apache Web Server
D. rlogin

Detect 4 – Connections to Port 5101

Here's an interesting scan that is still in progress at this very moment. I can't figure out what is going on. This scan started last Sept 21 and has been going on continuously ever since.

```
fw1.2001-0921-235901.log:21Sep2001 13:38:29 drop fw1 >at0
proto tcp src 137.140.59.14 9 dst 140.147.40.67 service 5101
s_port nimreg len 48 rule 130
fw1.2001-0921-235901.log:21Sep2001 18:54:56 drop fw1 >at0
proto tcp src 216.254.152.248 dst 140.147.40.67 service 5101
s_port 1320 len 48 rule 130
fw1.2001-0924-235900.log:24Sep2001 15:38:36 drop fw1 >at0
proto tcp src 64.50.148.188 dst mm6b4011 -140-147-23-150
service 5101 s_port 49817 len 48 rule 128
fw1.2001-0924-235900.log:24Sep2001 17:00:11 drop fw1 >at0
proto tcp src 138.88.36.84 dst mm6b4011 -140-147-23-150 service
5101 s_port 1024 len 44  rule 128
fw1.2001-0924-235900.log:24Sep2001 18:09:52 drop fw1 >at0
proto tcp src 137.140.58.175 dst 140.147.40.67 service 5101
s_port 1972 len 48 rule 128

ANY GOOD IDEAS OR GUESSES ABOUT WHAT MIGHT
BE GOING ON HERE?
```

Source of Trace

This trace was obtained from http://www.incidents.org/archives/intrusions/msg03818.html and was submitted to by Logan Choi on Thu, 14 Feb 2002.

Detect Was Generated By

CheckPoint Firewall-1. The log format for CheckPoint Firewall -1 is:

```
Date, Time, FW Action, FW Name, FW Interface, Protocol,
Src Address, Dst Address, Service, Src Port, Packet
Length, rule #
```

Probability the Source Address was Spoofed

The probability of spoofed source addresses in this instance is difficult to determine. The answer is dependant on the actual nature of the activity. If the traffic is genuine access of a valid service, then the likelihood of spoofed addresses is extremely low. However, if the activity is exploiting a vulnerability, then the likelihood of spoofed source addresses increases dramatically. Additionally, without information regarding the ttl it would be difficult to verify the genuiness of these addresses.

Description of Attack

There is not a lot of information to develop an accurate description of this attack. The main clue is the use of port 5101, which is used by the following services:

- Talarian, as listed by IANA, http://www.iana.org/assignments/port-numbers. According to the Talarian Web site, http://www.talarian.com/products/index.shtml :

  Talarian provides infrastructure software for connecting distributed applications and instantly exchan ging information across the enterprise or the Web.
- There is a pervasive computing program called "one.world" from University of Washington that uses port 5101 by default (according to Mike Poor's comments at http://www.incidents.org/archives/intrusions/msg03819.html). Their web page provides information regarding default ports used by this application http://one.cs.wash ington.edu/configuration.html .
- WinMx (napster like file sharing) servers originally used the napster range of ports around 6699 TCP and 6257 UDP, although this has changed (also according to Mike Poor's comments). http://www.devil-insi.de/pages/usefulports.html shows this usage.
- A search on Google revealed this web page, indicating that ICQ may also use port 5101 http://www.reuteras.com/Mutant_64/html/article.php?sid=64 .

Attack Mechanism

Since the source port is not constant and is always an ephemeral port, whereas the destination port is always 5105, it would seem we are observing a stimulation. Most likely the sour ce is attempting to reach a known service running on the destination. Without further knowledge of what service is using port 5101 on the two internal hosts, it is not clear if this is an attack, or simply accessing a service.

Correlations

Port 5101 has be en reported at Dshield every day over the period 2002-02-23 to 2002-03-26 (up to 152 times on 2002-03-09). However, no reports were made against any of the subnets containing the source hosts involved in this log targetting port 5101.

Evidence of Active ta rgetting

Because only 2 internal hosts have received packets for port 5101 it would seem as though the source knows what hosts are running the service they are targetting. Consequently, there is definite evidence of active targetting.

Severity

Target criticailty: 3

> With the information provided it is unkown what services are running on the hosts, but obviuosly the external source does.

Attack Lethality: 3

> Without knowing what service is being accessed on the internal hosts, there is no way of determining i f there are any vulnerabilities associated with them.

System Countermeasures: 2

> There appears to be a service running on the hosts that is known to the source and nothing is known of it's security.

Network Countermeasures: 5

Because the firewall has droppe d the packets, there is substantial network security to prevent a potential exploit of this service.

Attack Severity: (3 + 3) – (2 + 5) = -1

Defensive Recommendation

Investigate the two internal hosts 140.147.40.67 and 140.147.32.150 to see what service has opened this port. If it is not required, remove it. Otherwise ensure it is patched and there are no known vulnerabilities and allow it through the perimeter firewall.

Multiple Choice Test Question

Based on this log, what is the destination port:

```
fw1.2001-0921-235901.log:21Sep2001 13:38:29 drop fw1 >at0
proto tcp src 137.140.59.149 dst 140.147.40.67 service
5101 s_port nimreg len 48 rule 130
```

A. 130
B. 5101
C. nimreg
D. 49817

Detect 5 – Potential dtspcd Exploit

```
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.132:61 12
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.133:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.134:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.136:6112
SYN ******S*
Feb 11 01:19:5 9 204.192.116.243:6112 -> xxx.xxx.xxx.137:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.138:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.139:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.151:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.152:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.153:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.154:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.155:6112
SYN ******S*
```

```
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.156:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.158:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243: 6112 -> xxx.xxx.xxx.159:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.160:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.162:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.191:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.193:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.194:6112
SYN ******S*
Feb 11 01:19:59 204.192.116.243:6112  -> xxx.xxx.xxx.200:6112
SYN ******S*
Feb 11 01:19:59 204 .192.116.243:6112 -> xxx.xxx.xxx.201:6112
SYN ******S*
Feb 11 01:20:00 204.192.116.243:6112  -> xxx.xxx.xxx.64:6112
SYN ******S*
Feb 11 01:20:00 204.192.116.243:6112  -> xxx.xxx.xxx.65:6112
SYN ******S*
Feb 11 01:20:00 204.192.116.243:6112  -> xxx.xxx.xxx. 66:6112
SYN ******S*
Feb 11 01:20:00 204.192.116.243:6112  -> xxx.xxx.xxx.67:6112
SYN ******S*


Feb 10 22:35:59 216.38.192.53:6112  -> xxx.xxx.xxx.131:6112 SYN
******S*
Feb 10 22:35:59 216.38.192.53:6112  -> xxx.xxx.xxx.132:6112 SYN
******S*
Feb 10 22:35:59 216.38.192.53:6112  -> xxx.xxx.xxx.134:6112 SYN
******S*
Feb 10 22:35:59 216.38.192.53:6112  -> xxx.xxx.xxx.130:6112 SYN
******S*
Feb 10 22:35:59 216.38.192.53:6112  -> xxx.xxx.xxx.143:6112 SYN
******S*
Feb 10 22:35:59 216.38.192.53:6112  -> xxx.xxx.xxx. 153:6112 SYN
******S*
Feb 10 22:35:59 216.38.192.53:6112  -> xxx.xxx.xxx.152:6112 SYN
******S*
```

Source of Trace

This trace was obtained from

http://www.incidents.org/archives/intru sions/msg03765.html and was submitted by Ernie Pritchard on Mon, 11 Feb 2002

Detect Was Generated By

The detect appears to be generated by Snort running with the *–A fast* option set to minimise the output. Following is an explanation of the Snort logs in t his mode:

```
Date, Time, First.IP.Address:Port Number, ->(direction of data
flow arrow), Second.IP.Address:Port Number, Flags
```

Probability the Source Address was Spoofed

It is unlikely the address was spoofed. Otherwise no response would be returned to the attacker to inform him of what hosts were running the service they are targetting.

Description of Attack

According to Cert:

*Internet Security Systems (ISS) X-Force has reported a remotely exploitable buffer overflow in t he Common Desktop Environment (CDE) Subprocess Control Service (*dtspcd*). CDE is an integrated graphical user interface that runs on Unix and Linux operating systems.* dtspcd *is a network daemon that accepts requests from clients to execute commands and laun ch applications remotely. On systems running CDE,* dtspcd *is spawned by the Internet services daemon (typically* inetd *or* xinetd*) in response to a CDE client request.* dtspcd *is typically configured to run on port 6112/tcp with root privileges.* dtspcd *makes a function call to a shared library,* libDTSvc.so.1*, that contains a buffer overflow condition in the client connection routine. The buffer overflow can be exploited by a specially crafted CDE client request. Although the buffer overflow occurs in a shared l ibrary, the CERT/CC is not aware of any other CDE applications that use the vulnerable function.*

For more informatin see http://www.kb.cert.org/vuls/id/172583 and http://www.cert.org/advisories/CA -2002-01.html.

An alternative explanation is that “Diablo II” and “Starcraft” use port 6112 when played online http://advice.networkice/advice/exploits/ports/6112/default.html . However, David Anders’ explanation at http://www.incidents.org/archives/intrusions/msg03797.html probably rules that option out:

Port 6112/TCP (UDP as well) is a communications port for Blizzard's Battle.net gaming service used to play Diablo II and Starcraft online, but none of the source IP's noted correspond to Battle.net's address space.

I haven't heard about any re cent exploits for these applications that would explain the blanket syn scans.

If the attacker is scanning for existence of the *dtspcd* service, the next step would be to map which hosts are running the service. Then the attacker could attempt to exploit the known vunerability for this service, as described above.

Attack Mechanism

Either a TCP connect scan or a TCP SYN scan is being performed. A TCP connect scan is the most basic scan and is an attempt to establish a three -way handshake with the destination host on the chosen port. A TCP SYN scan is a litlle more stealthy. After the destination responds with a SYN and ACK, the source does not complete the three-way handshake with an ACK, leaving the connection half open. Some firewall and intrusion detection systems will not detect this sort of scan. Because this type of scan has become so common, most systems can detect it now.

Since the scan appears to be sccanning the class C address space in a somewhat random pattern, it has probably been performed with a port scanning tool, similar to Nmap or ScanPort. It is difficult to determine exactly which tool was used without more information, such as TCP sequence and ACK numbers and IP ID.

The matching source port and destination port are also evidence of packet cr afting using a port scanning tool. Normal IP behaviour is for the client host to chose a random ephemeral source port when it connects to the "well -known" port number of the server.

Correlations

Port 6112 has many reports against it recorded at dshield (http://www.dshield.org/port_report.php?port=6112&Submit=Submit+Query)

Dshield also reports a large number of records against both IP addresses which have launched this scan , but not targetting port 6112. Incidents.org lists several other detects of targetting port 6112 from a variety of source IP addresses. The following people report this attack also originating from 204.192.116.243:

- Michael Dwyer on Mon, 11 Feb 2002 - http://www.incidents.org/archives/intrusions/msg03766.html
- Chris Grout on Mon, 11 Feb 2002 - http://www.incidents.org/archives/intrusions/msg03768.html

The author received 6 alerts of this attack on his home PC while doing the final editing of this paper.

Evidence of Active targetting

Yes. The attacker is most likely searching for hosts with vulnerable version of *dtspcd* running.

Severity

Target criticality: 5

Any Unix or Linux host running *dtspcd* is potentially vulnerable

Attack lethality: 5

An attacker can execute arbitrary code with root privileges.

System countermeasures: 2

Since this is a recently discovered vulnerability the likelihood of systems being patched against it is low.

Network countermeasures: 2

Without knowing what perimiter defenses are in place, this is a guess. Based on the assumption that the intrusion detection system is located inside the

firewall, this would indicate the packets have already reached the internal network. However, if the intrusion detection system is located outside the firewall and the firewall is blocking traffic on port 6112, this rating could be increased.

Attack severity: (5 + 5) – (2 + 2) = 6

Defensive Recommendation

The Cert advisory recommends three approaches to protect against this vulnerability:

- Patch all systems running *dtspcd* with the appropriate patch from your vendor.
- Disable *dtspcd* until the patch can be appli ed.
- Block or restrict external access to port 6112.

For more information see http://www.kb.cert.org/vuls/id/172583 .

Multiple Choice Test Question

What is the most likely explanation for this trace?

```
Feb 10 22:35:59 216.38.192.53:6112  -> xxx.xxx.xxx.131:6112 SYN
******S*
Feb 10 22:35:59 216.38.192.53:6112  -> xxx.xxx.xxx.132:6112 SYN
******S*
Feb 10 22:35:59 216.38.192.53:6112  -> xxx.xxx.xxx.134:6112 SYN
******S*
Feb 10 22:35:59 216.38.192.53:6112  -> xxx.xxx.xxx.130:6112 SYN
******S*
Feb 10 22:35:59 216.38.192.53:6112  -> xxx.xxx.xxx.143:6112 SYN
******S*
Feb 10 22:35:59 216.38.192.53:6112  -> xxx.xxx.xxx.153:6112 SYN
******S*
Feb 10 22:35:59 216.38.192.53:6112  -> xxx.xxx.xxx.152:6112 SYN
******S*
```

A. Normal network traffic for the network game Diablo
B. SYN Flood
C. Reconnaissence scan looking for the *dtspcd* service
D. The source address was probably spoofed

Assignment 3 – "Analyse This" Scenario

Executive Summary

As requested, a security audit has been performed for your university network. Five consecutive days' of Snort data were provided for the period of Feb 15, 2002 to Feb 19, 2002.

Separate alert, scan and OOS (out -of-specification) logs were provided for each day. The log files used are listed in table 2, below:

Alerts	Scans	Out of Spec
alert.020215.gz	scans.020215.gz	oos_Feb.15.2002.gz
alert.020216.gz	scans.020216.gz	oos_Feb.16.2002.gz
alert.020217.gz	scans.020217.gz	oos_Feb.17.2002.gz
alert.020218.gz	scans.020218.gz	oos_Feb.18.2002.gz
alert.020219.gz	scans.020219.gz	oos_Feb.19.2002.gz

Table 2: Snort logs used for auditing

The analysis process is described in detail in Appendix A.

A list of all the detects reported by Snort is included. The 5 most prevelant detects have been reviewed in detail for the audit. A link graph has been produced for the OOS data to illustrate the relationship between the attackers and victims. Five external hosts were further investigated.

A total of 434,683 alerts of 84 different types, 2,596,771 scan s and 20 OOS packets were detected during the 5 days of the audit period.

It was observed that most of the traffic reported by Snort was internal to the university. However, for the *spp_http_decode: IIS Unicode attack detected* and the *MISC Large UDP Packet* a large amount of the traffic was between the university and various networks located in Korea. It would be advisable to investigate any relationships between the users of the internal hosts and Korea.

A very high proportion of the alerts involved hosts f rom the internal subnets MY.NET.150.0, MY.NET.151.0, MY.NET.152.0 and MY.NET.153.0. The recommendation of this audit would be to determine whether this traffic is legitimate and take appropriate defensive measures if not.

Alerts Detected

A total of 434,683 alerts of 84 different types were detected during the period of Feb 15, 2002 to Feb 19, 2002. Table 3, below, lists all the alerts, sorted by number of occurrences. The first 5 types of alerts are described in detail below.

Frequency	Description
connect to 515 from inside	145159
spp_http_decode: IIS Unicode attack detected	76041
SMB Name Wildcard	61513
MISC Large UDP Packet	34880

ICMP Echo Request L3retriever Ping	30434
spp_http_decode: CGI Null Byte attack detected	20467
INFO MSN IM Chat data	13009
SNMP public access	12982
High port 65535 udp - possible Red Worm - traffic	8380
Watchlist 000220 IL-ISDNNET-990517	7087
ICMP Echo Request Nmap or HPING2	4701
ICMP Echo Request BSDtype	2944
ICMP Fragment Reassembly Time Exceeded	2730
MYPARTY - Possible My Party infection	1888
ICMP Router Selection	1570
WEB-IIS view source via translate header	1541
FTP DoS ftpd globbing	1357
INFO Inbound GNUTella Connect request	954
ICMP Echo Request Delphi-Piette Windows	613
Watchlist 000219	609
SCAN Proxy attempt	588
NMAP TCP ping!	527
ICMP Destination Unreachable (Host Unreachable)	502
INFO - Possible Squid Scan	495
INFO Possible IRC Access	374
WEB-IIS _vti_inf access	374
WEB-FRONTPAGE _vti_rpc access	359
WEB-CGI ksh access	324
Null scan!	293
WEB-CGI scriptalias access	259
INFO Outbound GNUTella Connect request	243
INFO FTP anonymous FTP	182
INFO Napster Client Data	163
ICMP Echo Request Windows	129
Incomplete Packet Fragments Discarded	107
ICMP traceroute	97
ICMP Destination Unreachable (Commu nication	79
IDS552/web-iis_IIS ISAPI Overflow ida nosize	72
WEB-CGI csh access	70
WEB-MISC Attempt to execute cmd	60
INFO Inbound GNUTella Connect accept	47
WEB-MISC compaq nsight directory traversal	38
WEB-MISC 403 Forbidden	36
EXPLOIT x86 setgid 0	32
Possible trojan server activity	28
Back Orifice	27
Attempted Sun RPC high port access	26
WEB-CGI phf access	26
SCAN Synscan Portscan ID 19104	25
Watchlist 000222 NET-NCFC	21
MISC traceroute	20
INFO Outbound GNUTella Connect accept	17

Tiny Fragments - Possible Hostile Activity	16
Port 55850 tcp - Possible myserver activity - ref.	15
EXPLOIT NTPDX buffer overflow	14
ICMP Destination Unreachable (Protocol Unreachable)	13
FTP CWD / - possible warez site	13
Queso fingerprint	13
SMB CD...	13
EXPLOIT x86 NOOP	11
EXPLOIT x86 stealth noop	10
SCAN FIN	9
EXPLOIT x86 setuid 0	7
High port 65535 tcp - possible Red Worm - traffic	7
RFB - Possible WinVNC - 010708-1	6
SUNRPC highport access!	5
Port 55850 udp - Possible myserver activity - ref.	4
WEB-CGI formmail access	4
WEB-MISC ICQ Webfront HTTP DOS	3
WEB-MISC http directory traversal	3
Virus - Possible MyRomeo Worm	2
WEB-MISC whisker head	2
Fragmentation Overflow Attack	2
TCP SRC and DST outside network	2
X11 outgoing	1
DNS named iquery attempt	1
BACKDOOR NetMetro File List	1
WEB-IIS encoding access	1
TFTP - External UDP connection to internal tftp server	1
TFTP - Internal UDP connection to external tftp server	1
Virus - Possible pif Worm	1
NIMDA - Attempt to execute cmd from cam pus host	1
x86 NOOP - unicode BUFFER OVERFLOW ATTACK	1
WEB-IIS asp-dot attempt	1

Table 3: Alerts sorted by number of occurrences

Connect to 515 From Inside

The top 5 source IPs are listed in table 4, below:

Source IP	No of Alerts	No Destination IP
MY.NET.153.119	11981	1
MY.NET.153.114	10342	1
MY.NET.153.122	7724	1
MY.NET.153.109	6927	1
MY.NET.153.126	6198	1

Table 4: Top 5 source IPs for "Connect to 515 from inside"

All the destination IPs are listed in table 5, below:

Destination IP	No of Alerts	No Source IPs
MY.NET.150.198	144777	141
MY.NET.1.63	195	5
MY.NET.153.184	187	1

Table 5: Destination IPs for "Connect to 515 from inside"

Sample Trace

```
02/15-07:34:25.996344  [**] conn ect to 515 from inside [**]
MY.NET.153.114:2657 -> MY.NET.150.198:515
02/15-07:34:25.996412  [**] connect to 515 from inside [**]
MY.NET.153.114:2657 -> MY.NET.150.198:515
02/15-07:34:25.996932  [**] connect to 515 from inside [**]
MY.NET.153.114:2657 -> MY.NET.150.198:515
02/15-07:34:25.996999  [**] connect to 515 from inside [**]
MY.NET.153.114:2657 -> MY.NET.150.198:515
02/15-07:34:25.999621  [**] connect to 515 from inside [**]
MY.NET.153.114:2657 -> MY.NET.150.198:515
```

Brief Description

This detect indicates an attacker attempting to exploit one of the following vulnerable services operating on port 515:

- A buffer overflow in the Solaris line printer daemon (*in.lpd*) that may allow a remote intruder to gain root privileges or crash the printer daemon.
- *LPRng* (common in many open source systems) has a missing format string argument in at least two calls to the *syslog()* function. A remote user may be able to execute arbitrary code with elevated privileges. In addition, the printing service may be disrupted or disabled entirely.

Snort detected 145,159 alerts of port 515 being scanned from hosts within the univesity. These alerts were generated from 147 different source IPs, but directed at just 3 destinations. The host MY.NET.150.198 was the target of the vast majority of the scans, being scanned by most of the hosts on the MY.NET.152 and MY.NET.153 subnets. This may indicate that all (or at least most) of the hosts on these 2 subnets have been compromised. All the 195 scans against MY.NET.1.63 originated from 5 separate hosts all within the MY.NET.149 subnet possibly indicating internal users trying to exploit this host. Finally, MY.NET.153.184 received 187 scans, all from MY.NET.60.8 which was also one of the hosts responsible for scanning MY.NET.150.98. This m ay indicate that MY.NET.60.8 could be the host responsible for controlling all the potentially compromised hosts in MY.NET.152 and MY.NET.153.

Defensive Recommendation

It is highly recommended the Administrator patch the following three hosts: MY.NET.150.198, MY.NET.1.63 and MY.NET.153.184 for their respective vulnerability.

I would advise investigating MY.NET.60.8 to confirm if it is being used legitimately. It would also be very wise to check all the hosts in the MY.NET.152 and MY.NET.153 subnets for signs of compromise.

Correlation

No correlations could be found with any previous students' practicals.

ssp_http_decode: IIS Unicode Attack Detected

The top 5 internal source IPs are listed in table 6, below:

Source IP	No of Alerts	Destination IPs
MY.NET.153.143	4522	48
MY.NET.153.110	4304	74
MY.NET.153.197	3625	75
MY.NET.153.147	3210	53
MY.NET.153.106	3175	76

Table 6: Top 5 internal source IPs for "ssp_http_decode: IIS Unicode Attack Detected"

The external source IPs are listed in table 7, below:

Source IP	No of Alerts	Destination IP
203.227.74.100	2	1
200.64.239.185	6	1
195.192.126.34	2	1
130.37.130.39	4	1
80.129.159.77	6	6

Table 7: Esternal source IPs for "ssp_http_decode: IIS Unicode Attack Detected"

The external destination IPs are listed in table 8, below:

Destination IP	No of Alerts	Source IP
211.115.213.202	5701	13
211.111.220.163	4393	5
211.115.213.207	2414	12
211.111.214.125	2165	1
211.233.29.216	1554	15

Table 8: Top 5 exteral destinations for "ssp_http_decode: IIS Unicode Attack Detected"

The top 5 internal destination IPs are listed in table 9, below:

Destination IP	No of Alerts	Source IP
MY.NET.5.96	14	6
MY.NET.5.241	5	2
MY.NET.11.4	4	4

MY.NET.253.114	3	1
MY.NET.5.97	1	1

Table 9: Top 5 internal destinations for "ssp_http_decode: IIS Unicode Attack Detected"

Brief Description

As with the *connect to 515 from inside* alerts, most of these 76,041 alerts were generated by hosts on the MY.N ET.152 and MY.NET.153 subnets. The subnet 211.0.0.0 was the most common destination, specifically 211.32.0.0 and 211.233.0. These subnets are located in Korea.

As can be seen in the first sample of this alert, it was generally associated with *WEB-MISC Attempt to execute cmd* alerts for traffic entering the university network.

Sample Trace

```
02/15-05:42:07.645222  [**] WEB -MISC Attempt to execute cmd
[**] 130.37.130.39:26336  -> MY.NET.5.241:80
02/15-05:42:08.006765  [**] WEB -MISC Attempt to execute cmd
[**] 130.37.130.39:26355  -> MY.NET.5.241:80
02/15-05:42:08.347371  [**] WEB -MISC Attempt to execute cmd
[**] 130.37.130.39:26378  -> MY.NET.5.241:80
02/15-05:42:08.706636  [**] WEB -MISC Attempt to execute cmd
[**] 130.37.130.39:26406  -> MY.NET.5.241:80
02/15-05:42:09.077616  [**] WEB -MISC Attempt to execute cmd
[**] 130.37.130.39:26420  -> MY.NET.5.241:80
02/15-05:42:09.790498  [**] spp_http_decode: IIS Unicode
attack detected [**] 130.37.130.39:26453  -> MY.NET.5.241:80
02/15-05:42:09.790498  [**] WEB -MISC Attempt to execute cmd
[**] 130.37.130.39:26453  -> MY.NET.5.241:80
02/15-05:42:10.125567  [**] spp_http_decode: IIS Unicode
attack detected [**] 130.37.130.39:26468  -> MY.NET.5.241:80
02/15-05:42:10.125567  [**] WEB -MISC Attempt to execute cmd
[**] 130.37.130.39:26 468  -> MY.NET.5.241:80
02/15-05:42:10.486039  [**] spp_http_decode: IIS Unicode
attack detected [**] 130.37.130.39:26480  -> MY.NET.5.241:80
02/15-05:42:10.486039  [**] WEB -MISC Attempt to execute cmd
[**] 130.37.130.39:26480  -> MY.NET.5.241:80
02/15-05:42:10.847752  [**] spp_http_decode: IIS Unicode
attack detected [**] 130.37.130.39:26499  -> MY.NET.5.241:80
02/15-05:42:10.847752  [**] WEB -MISC Attempt to execute cmd
[**] 130.37.130.39:26499  -> MY.NET.5.241:80
02/15-05:42:11.194714  [**] WEB -MISC Attempt to execute cmd
[**] 130.37.130.39:26517  -> MY.NET.5.241:80
02/15-05:42:11.541634  [**] WEB -MISC Attempt to execute cmd
[**] 130.37.130.39:26533  -> MY.NET.5.241:80
02/15-05:42:11.905875  [**] WEB -MISC Attempt to execute cmd
[**] 130.37.130.39:26550  -> MY.NET.5.241:80
```

```
02/15-05:42:12.257921  [**] WEB-MISC Attempt to execute cmd
[**] 130.37.130.39:26567  -> MY.NET.5.241:80
```

This attack exploits the directory traversal vulnerability in Microsoft IIS 4.0, 5.0 and Windows98 Personal Web server. The attacker can gain access to any file on the local logical drive the *IUSR_machinename* account has rights to. By default this account is a member of the Everyone and Users groups. This vunerability is commonly exploited by the Code Blue worm. For more detailed information see http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=1806 :

In the following sample each alerts was generated 3 times with identical time stamps. This was by far the most common occurrence with this alert (on some occassions up to 9 alerts with identical timestamps occurring), but only when the traffic was leaving the unversity network. The best explanation for this is that Snort detected the same event with separate rules and therefore may be alerting on normal behaviour.

Sample Trace

```
02/15-07:32:05.522894  [**] spp_http_decode: IIS Unicode
attack detected [**] MY.NET.150.97:2650  -> 207.200.89.193:80
02/15-07:32:05.522894  [**] spp_http_decode:  IIS Unicode
attack detected [**] MY.NET.150.97:2650  -> 207.200.89.193:80
02/15-07:32:05.522894  [**] spp_http_decode: IIS Unicode
attack detected [**] MY.NET.150.97:2650  -> 207.200.89.193:80
02/15-07:32:05.933967  [**] spp_http_decode: IIS Unicode
attack detected [**] MY.NET.150.97:2651  -> 207.200.89.193:80
02/15-07:32:05.933967  [**] spp_http_decode: IIS Unicode
attack detected [**] MY.NET.150.97:2651  -> 207.200.89.193:80
02/15-07:32:05.933967  [**] spp_http_decode: IIS Unicode
attack detected [**] MY.NET.150.97:2651  -> 207.200.89.193:80
```

On closer inspection, this problem is likely much smaller than it initially appeared. Only 20 of the alerts were generated from external hosts accessing the university network.

Defensive Recommendation

My advise would be to check the Snort rules in use and try to eliminate multiple detects for this event on outgoing traffic.

The most effective defense against this exploit is to ensure all Microsoft IIS 4.0, 5.0 and Windows 98 Personal Web Servers are patched. It would be wise to remove these services from hosts that do not require them, according to the university's policy. Running reliable anti-virus software that is kept up to date would also help minimise the risk.

Correlation

No correlations could be found with any previous students' practicals or on DShield.

SMB Name Wildcard

The top 5 internal source IPs are listed in table 10, below:

Source IP	No of Alerts	No Destination IP
MY.NET.11.6	13808	59
MY.NET.11.7	11900	59
MY.NET.11.5	4040	59
MY.NET.152.163	1119	3
MY.NET.152.167	734	4

Table 10: Top 5 internal source IPs for "SMB Name Wilcard"

The external source IPs are listed in table 11, below:

Source IP	No of Alerts	No Destination IP
169.254.22.29	19	9
67.32.185.14	1	1

Table 11: External source IPs for "SMB Name Wilcard"

The top 5 destination IPS are listed in table 12, below:

Destination IP	No of Alerts	Source IP
MY.NET.11.6	13711	59
MY.NET.11.7	11865	59
MY.NET.11.5	4053	59
MY.NET.152.163	1122	3
MY.NET.152.167	742	5

Table 12: Top 5 destination IPs for "SMB Name Wilcard"

Sample traces

```
02/15-00:00:07.544703  [**] ICMP Echo Request L3retriever Ping
[**] MY.NET.152.249 -> MY.NET.11.6
02/15-00:00:07.544878  [**] SMB Name Wildcard [**]
MY.NET.152.249:137 -> MY.NET.11.6:137
02/15-00:00:07.545163  [**] SMB Name Wildcard [**]
MY.NET.11.6:137 -> MY.NET.152.249:137

02/15-00:01:09.893286  [**] ICMP Echo Request Nmap or HPING2
[**] MY.NET.152.213 -> MY.NET.11.6
02/15-00:01:12.325559  [**] SMB Name Wild card [**]
MY.NET.11.6:137 -> MY.NET.152.213:137
```

Brief Description

This is potentially part of a scan looking for Windows systems or Linux systems running the Samba service with unprotected network shares. The attacker may be aiming to compromise the system s, or it may be a worm looking for a means of propogation.

Only one of these 61513 alerts originated from an IP address outside the university (67.32.185.14). The remainder were either from MY.NET.x.x addresses or from one of the default addresses assigned when no DHCP server is available (169.254.22.29). This is normal behaviour for a network running NetBIOS NameSevices on port 137.

However, this alert is made more intersting due to its associations. For each internal occurrence of this event, it was alway s appearing in one of 2 patterns as shown in the sample traces above. This likely indicates it is associated with some form of network scanning. The "ICMP Echo Request L3Retriever Ping" is explained in detail later. The "ICMP Echo Request Nmap or HPING2" i s not covered in this paper.

Defensive Recommendation

Since the problem is primarily an internal one, is would be safe to assume that perimiter security is in place to prevent NetBIOS traffic from entering and leaving the network. The advise would be to di sable Windows networking shares or preferably NetBIOS on the Microsoft hosts and Samba on the Linux hosts if sharing is not required, in line with the university's policy. For those hosts that do need to share files, ensure strong passwords are used to pro tect the shares.

Correlation

No correlations could be found with any previous students' practicals.

MISC Large UDP Packet

The top 5 source IPs are listed in table 13, below:

Source IP	No of Alerts	Destination IPs
209.177.65.18	7749	2
63.240.15.205	5444	1
216.54.221.197	4759	1
210.220.161.101	3817	1
63.240.15.204	2471	1

Table 13: Source IPs for "Misc large UDP packet"

The top 5 destination IPs are listed in table 14, below:

Destination IP	No of Alerts	Source IPs
MY.NET.153.197	11383	5
MY.NET.152.163	5366	1
MY.NET.153.171	4759	1
MY.NET.152.12	4201	2
MY.NET.152.168	2383	1

Table 14: Destination IPs for "Misc large UDP packet"

Sample Trace

02/15-08:32:14.718034 [**] MISC Large UDP Packet [**] 211.233.70 .162:3948 -> MY.NET.153.196:1612

02/15-12:30:56.338359 [**] MISC Large UDP Packet [**] 61.78.53.74:4543 -> MY.NET.153.165:1323

02/15-12:30:56.463555 [**] MISC Large UDP Packet [**] 61.78.53.74:4543 -> MY.NET.153.165:1323

02/15-12:30:56.672696 [**] MIS C Large UDP Packet [**] 61.78.53.74:4543 -> MY.NET.153.165:1323

02/15-12:30:56.745068 [**] MISC Large UDP Packet [**] 61.78.53.74:4543 -> MY.NET.153.165:1323

Brief Description

This alert was detected 34,880 times over the 5 day period analysed.

It is possible to use UDP packets for DOS attacks. Since only one host is being targetted at a time this is one possible explanation of this attack. However the packet rate never seems to exceed about 10 per second, probably averaging about 6 -8 per second, thus eliminating that theory. Since the packets are UDP and no acknowledgement is expected, there is a high probability the source IP addresses are spoofed.

As with the other alerts discussed previously, the internal subnets MY.NET.152.0 and MY.NET.153.0 are the mo st effected by this attack. It is also interesting to note that 1,972 of these alerts had a source port of 0 and of those, 1,938 also had a destination port of 0. Many other alerts detected many successive packets from the same host without the source port changing. Both these observations are likely indicators of crafted packets. Some of these attacks appear to involve 3 or 4 hosts from the same external subnet targetting one host within the university.

Defensive Recommendation

Block UDP at the perimiter f irewalls except for the ports required by approved services. Disable unnecessary services on hosts.

Correlation

No correlations were found for the source IP addresses with previous students' practicals. Rick Yuen (http://www.giac.org/practical/Rick_Yuen_GCIA.doc) reported other destinations within the MY.NET.153.0 subnet (MY.NET.153.187, MY.NET.153.185 and MY.NET.153.149) being targetted in the same way.

ICMP Echo request L3Retriever Ping

Brief Description

The top 5 source Ips are listed in table 15, below:

Source IP	No of Alerts	Destination IP
MY.NET.152.163	1110	3
MY.NET.152.167	748	3
MY.NET.152.180	647	3
MY.NET.152.183	627	3

MY.NET.152.162	621	3

Table 15: Source IPs for "ICMP Echo Request L3Retriever Ping"

The top 5 internal destination IPs are listed in table 16, below:

Destination IP	No of Alerts	Source IP
MY.NET.11.6	13796	59
MY.NET.11.7	11887	59
MY.NET.11.5	4059	59
MY.NET.5.4	367	21
MY.NET.150.133	161	3

Table 16: Internal destination IPs for "ICMP Echo Request L3Retriever Ping"

The external destination IP is listed in table 17, below:

Destination IP	No of Alerts	Source IP
216.32.244.30	4	1

Table 17: External destination IP for "ICMP Echo Request L3Retriever Ping"

These alerts are the result of scans performed by L -3 Network Security's Retriever (which has now been bought out by Symantec). For more information on this product see: http://www.symantec.com/press/security/n990525_ns.html .

Yet again, the MY.NET.150.0, MY.NET.151.0 and MY.NET.152.0 subnets featured largely in these events. Also, the hosts MY.NET.11.5, MY.NET.11.6 and MY.NET.11.7 were the key targets.

As highlighted under the "SMB Name Wildcard" description, the sample traces indicate it occurs in conjunction with *SMB Name Wildcard* alerts when both source and destination are internal:

Sample Trace

```
02/15-00:00:07.544703  [**] ICMP Echo Request L3retriever Ping
[**] MY.NET.152.249 -> MY.NET.11.6
02/15-00:00:07.544878  [**] SMB Name Wildcard [**]
MY.NET.152.249:137 -> MY.NET.11.6:137
02/15-00:00:07.545163  [**] SMB Name Wildcard [**]
MY.NET.11.6:137 -> MY.NET.152.249:137
```

This correlation doesn't exist when either of the hosts are external:

Sample Trace

```
02/19-17:15:26.062308  [**] ICMP Echo Request L3retriever Ping
[**] MY.NET.150.103 -> 216.32.244.30
02/19-17:15:28.412329  [**] ICMP Echo Request L3retriever Ping
[**] MY.NET.150.103 -> 216.32.244.30
```

Defensive Recommendation

Investigate the hosts on the MY.NET.150.0, MY.NET.151.0 and MY.NET.152.0 subnets (especially those listed in the top 5 source hosts) for existence of the L -3

Retriever software and verify whether it is being u sed legitimately. If not remove it from those hosts.

Correlation

No correlations could be found with any previous students' practicals.

OOS Data Link Graph

MY.NET.150.133 was the main target for OOS packets, receiving 10 from 7 different hosts. All These p ackets were aimed at TCP port 1214 (kazaa). A variety of flags were set as is shown in figure 5, below.

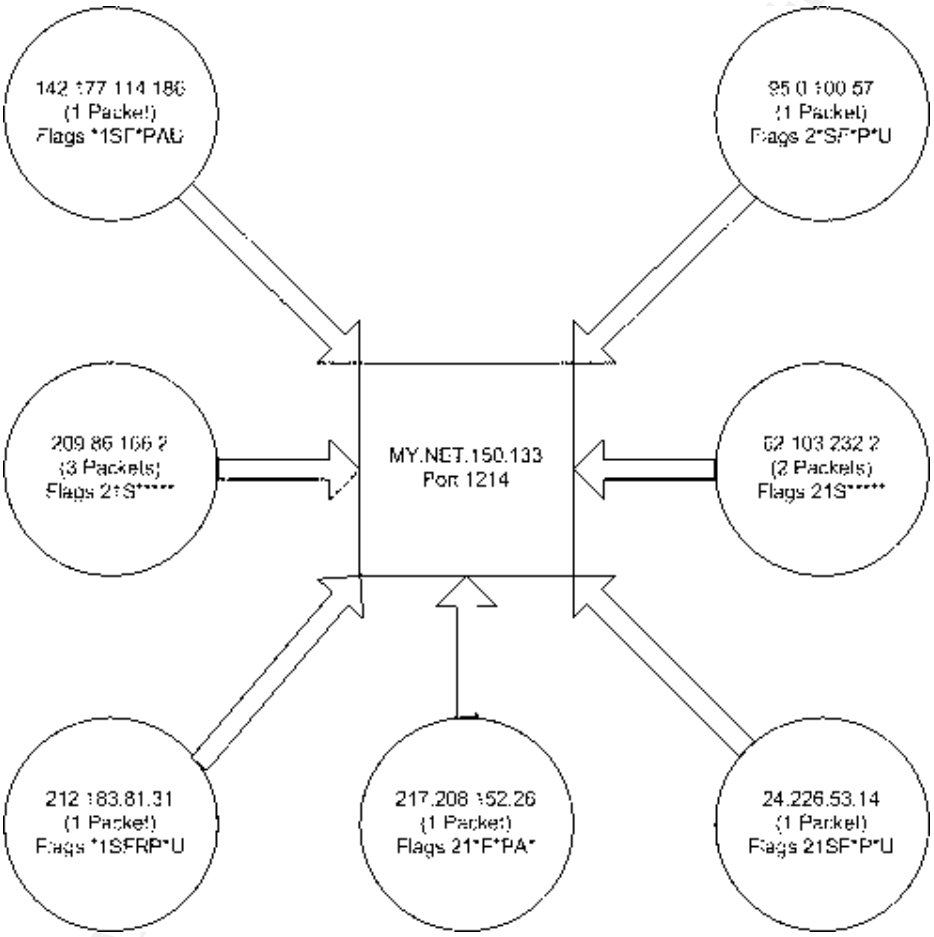

Figure 5: OOS Data Link Graph

Top Talkers

Below are lists of the top 10 talkers in various categories:

Source	No.
MY.NET.60.43	434257
MY.NET.6.49	175131
MY.NET.6.45	146905
MY.NET.6.52	112106
MY.NET.6.50	110673
MY.NET.6.48	109161

MY.NET.152.22	57552
MY.NET.60.11	53592
MY.NET.6.60	51600
MY.NET.6.53	48274

Table 18: Top 10 Internal sources

Source	No.
205.188.228.1	15035
63.215.70.141	13182
205.188.228.65	12688
205.188.228.17	12665
205.188.228.33	10514
209.177.65.18	7749
63.240.15.205	5491
216.54.221.197	5108
212.179.35.118	4647
210.220.161.101	4594

Table 19: Top 10 External Sources

Destination	No.
MY.NET.150.198	148075
MY.NET.1.7	113599
MY.NET.1.3	90491
MY.NET.11.6	77461
MY.NET.11.7	68106
MY.NET.1.4	62323
MY.NET.6.45	53969
MY.NET.60.43	43228
MY.NET.153.197	33125
MY.NET.6.60	31675

Table 20: Top 10 Internal Destinations

Destination	No.
209.10.239.135	18820
131.118.254.39	7739
205.188.228.17	6699
205.188.228.65	6624
131.118.254.38	6290
211.115.213.202	5739
205.188.228.1	4834
205.188.228.33	4681
211.111.220.163	4410
205.188.137.79	3629

Table 21: Top 10 External Destinations

Five External Sources for Further Investigation

The first four external IP addresses were chosen for further investigation because they represented the four top -talking subnets betwe en source and destination

addresses. The subnet 205.188.228.0 appeared 5 ftimes between these two lists. The fifth address was chosen since it was the top -talker outside of the United States.

209.10.239.135

Server used for this query: http://www.arin.net/ whois

```
Globix Corporation ( NETBLK-GLOBIXBLK3)
   295 Lafayette St - 3rd Fl
   NY, NY 10012
   US

   Netname: GLOBIXBLK3
   Netblock: 209.10.0.0 - 209.11.223.255
   Maintainer: PFMC

   Coordinator:
      Hostmaster, Globix Corporation  ( GCH2-ARIN)  arin-
admin@GLOBIX.NET
      +1-212-334-8500 (FAX) 212.334.8615

   Domain System inverse mapping provided by:

   Z1.NS.NYC1.GLOBIX.NET   209.10.66.55
   Z1.NS.SJC1.GLOBIX.NET   209.10.34.55
   Z1.NS.LHR1.GLOBIX.NET   212.111.32.38

   ADDRESSES WITHIN THIS BLOCK ARE NON -PORTABLE

   Record last updated on 05 -Apr-2001.
   Database last updated on  22 -Mar-2002 19:57:54 EDT.
```

205.188.228.1

Server used for this query: http://www.arin.net/whois

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

   Netname: AOL-DTC
   Netblock: 205.188.0.0 - 205.188.255.255

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

```
   703-265-4670

Domain System inverse mapping provided by:

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

Record last updated on 27-Apr-1998.
Database last updated on  22-Mar-2002 19:57:54 EDT.
```

63.215.70.141

Server used for this query: http://www.arin.net/whois

```
Streaming Media Corporation  (NETBLK-NETBLK-STRM8)
   6446 South Kenton Street, Suite 130
   Englewood, CO 80111
   US

   Netname: NETBLK-STRM8
   Netblock: 63.215.70.0 - 63.215.70.255

   Coordinator:
      Hostmaster, SMC  (ZH58-ARIN)  hostmaster@smc.net
      720-875-0700

   Record last updated on 13-Jun-2001.
   Database last updated on  22-Mar-2002 19:57:54 EDT.
```

209.177.65.18

Server used for this query: http://www.arin.net/whois

```
NCI Technologies, Inc. (NETBLK-NCI-BLK-1)
   PO Box 376
   Philipsburg, PA 16866
   US

   Netname: NCI-BLK-1
   Netblock: 209.177.64.0 - 209.177.95.255
   Maintainer: NCIT

   Coordinator:
      Bezilla, Daniel B.  (DB1208-ARIN)  dan@NCITECH.COM
      +1-814-342-7030 ext. 7102 (FAX) 81 4-342-7033
```

```
        Domain System inverse mapping provided by:

        NS3.NETPHD.NET          209.177.64.10
        NS2.NETPHD.NET          209.177.66.19

        ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

        Record last updated on 06-Jun-2001.
        Database last updated on  22-Mar-2002 19:57:54 EDT.
```

211.115.213.202

Server used for this query: http://www.whois.nic.or.kr/english

```
Korea Internet Information Service V1.0 ( created by KRNIC,
2001.6 )
query: 211.115.213.202

# ENGLISH

IP Address          : 211.115.213.0 -211.115.213.255
Network Name        : GNG-IDC-ILOVESCHOOL
Connect ISP Name    : GNGIDC
Connect Date        : 20001125
Registration Date   : 20010621

[ Organization Information ]
Orgnization ID      : ORG215464
Org Name            : iloveschool
State               : SEOUL
Address             : 724 Suseo-Dong Gangnam-Gu
Zip Code            : 135-934

[ Admin Contact Information]
Name                : Yungsuk Cho
Org Name            : iloveschool
State               : SEOUL
Address             : 724 Suseo-Dong Gangnam-Gu
Zip Code            : 135-934
Phone               : +82-2-538-0629
Fax                 : +82-2-3420-2301
E-Mail              : taiwa@iloveschool.co.kr

[ Technical Contact Information ]
```

Name : Yungsuk Cho
Org Name : iloveschool
State : SEOUL
Address : 724 Suseo -Dong Gangnam -Gu
Zip Code : 135 -934
Phone : +82 -2-538-0629
Fax : +82 -2-3420-2301
E-Mail : taiwa@iloveschool.co.kr

Appendix A

Ananlysis Process for Assignment 3 – "Analyse This"

Tools Used for Analysis

- Grep (Windows version)
- Sed (Windows version)
- Active Perl (Windows version)
- Microsoft Access 2000
- Microsoft Excel 2000
- Microsoft Visio 2002

Web Sites Used for Whois and NSLookup

- http://www.arin.net/whos/index.html
- http://www.ripe.net
- http://whois.apnic.net
- http://whois.nic.or.kr/english
- http://www.amnesi.com/hostinfo/ipinfo
- http://www.samspade.org/
- http://www.dshield.org

Web Sites Used for Alert Descriptions

- http://www.sans.org
- http://www.snort.org
- http://www.cert.org
- http://cve.mitre.org
- http://www.securityfocus.com

The following Internet search engines were used also:

- http://www.google.com

We Sites Used to Look Up Port Numbers

- http://www.snort.org/ports.html
- http://www.iana.org/assignm ents/port-numbers
- http://advice.networkice.com/advice/exploits/ports

Published References

- Northcutt, Stephen, Cooper, Mark, Fearnow ,Matt and Frederick, Karen, "Intrusion Signatures and A nalysis", New Riders Publishing, January 2001
- Cole, Eric, "Hackers Beware", New Riders Publishing, August 2001
- Northcutt, Stephen and Novak, Judy, "Network Intrusion Detection, An Analysts Handbook", 2 nd Edition, New Riders Publishing, 2000

- The SANS Institute, "TCP/IP for Intrusion Detection and Firewalls", course reference, Capitol SANS
- The SANS Institute, "Network Traffic Analysis Using TCPDump", course reference, Capitol SANS
- Stevens, W. Richard "TCP/IP Illustrated, Volume 1" Addison -Wesley, 1994

Alert Files

The alert files *alert.020215.clean* through *alert.020219.clean* were merged together into one file called *alertraw.txt.* Then I used Alan Woodroffe's (http://www.sans.org/y2k/practical/Alan_Woodroffe_GCIA.doc) *alert.sed* file altered to suit the Windows version of *sed.exe* to remove all the portscan entries and to convert the file into a semi-colon seperated file with the following fields:

date, time, fraction, source ip, source port , dest ip, dest port, description

This was performed with following command:

```
grep ^[01][0-9]/[0-9][0-9]- alertraw.txt | grep -v
     spp_portscan | sed -f alert.sed > alertout.txt
```

Semi-colons used as the delimiters since some of the description fields contained commas in the text. The resulting file was then imported into Microsoft Access. Various reports and queries were written to manipulate the data into the formats required for this audit.

Scans Data

The scans files *scans.020215.clean* through *scans.020219.*clean were also merged together into a filed called scansraw.txt. Again, Alan Woodroffe's *scans.sed* was altered for Windows to convert the file into a semi-colon seperated file with the same fields as for the alerts data. The following command was used in this instance:

```
grep ^[A-Z][a-z][a-z]. scansraw.txt | sed -f scans.sed >
     scansout.txt
```

To produce the top talkers list, the *Source IP* and *Destination IP* fields from both *alertout.txt* and *scansout.txt* files were imported into a second Access database and the tables were merged using a merge query.

This second database was created to be a more manageable size, speeding up the process of deteriming the top talkers.

OOS Data

Due to the small number of OOS packets in the 5 days audited, this data was analysed manually. The following fields were copied into a Microsoft Excel 2000 spreadsheet named *OOS_Data.xls*:

Source ip, Source port, Dest ip, Dest port, Flags

The data was then sorted first by *Dest IP*, then by *Dest Port*, then by *Source IP*. The resulting link graph was drawn in Microsoft Visio 2002 and saved as JPEG file named *OOS_Link.jpg*. This file was then embedded into this Word document.

If, however, a larger number of OOS packets were to be analysed, a more automated process, similar to those used for the Alerts and Scans data would save much time.