Global Information Assurance Certification Paper

GIAC Intrusion Detection In Depth
GCIA Practical Assignment v3.3

Robin Stuart, GSEC

October 24, 2002

TABLE OF CONTENTS

Assignment #1 - The State of Intrusion Detection

Caveat Analyst: Why Attack Simulation Software May Be Harmful Rather Than Helpful To The Signature Validation Process

Introduction

Functionally, the most common intrusion detection systems ("IDS's") detect two types of behaviors, anomalous activity and signature-based attacks. Attack "signatures" are a defined set of criteria, packet header features or network traffic patterns that match the characteristics of known malicious techniques or system vulnerabilities. A signature-based intrusion detection system is only as effective as the signatures applied. Therefore, it is incumbent upon the intrusion detection analyst to validate each signature. In order to do that, knowledge of exploit methodology is required. Vulnerability assessment scanners, such as Nessus, and intrusion detection signature validation programs, such as Blade Informer, ostensibly make the security analyst's life easier - or do they?

I submit that such programs may actually add a layer of complexity to signature testing and validation. Not only does the analyst require knowledge of the mechanics of the exploits and vulnerabilities which the IDS is guarding against, s/he also must become intimately familiar with the attack simulation program in use. The analyst needs to know how to use it effectively and be familiar with the particulars of its behavior; does it replicate the vulnerability or exploit exactly? If not, how does the program differ from an actual attack? Are the results of signature testing with the attack simulation product consistent, in other words, will the same attack produce the same results every time? Are the results of the simulated attack identical to the results of an actual attack? To answer these questions, a second method of signature testing should also be employed, whether it entails performing the actual exploit or running a second attack simulation program. The latter option brings the analyst to the same questions that the first method raises.

Exploit and signature updates are also an issue. Every time a new exploit or variation on an existing exploit emerges, the analyst must update the signatures on the IDS as well as the attack simulation program. Before the new signature can be tested, the analyst must test the simulation program to ensure that it does, indeed, simulate the new attack. In essence, the analyst must validate the validation method before employing it with confidence. You see where I'm heading with this - signature test methodology itself potentially becomes a full time job when using an attack simulation program purporting to make an analyst's life easier.

The Exploit

To illustrate my point, I call upon the FTP Bounce exploit. Although relatively simple in practice, the FTP Bounce involves three machines, the attacker, the victim and the unwitting accomplice, thus making it potentially more complex to detect. Perfect to display any inconsistencies between simulated attack methods as compared to signature validation using the actual exploit.

The exploit itself highlights a weakness in the FTP protocol which allows connections from anywhere to anywhere using the PORT command. Under the FTP protocol as defined by RFC 959, the client controls the data connection. The client opens a connection from its own ephemeral port to the FTP server's port 21. This is the control connection from which the client tells the server where to send data that it requests. In order to receive data, the client opens a second connection, known as the data connection, on the server using the PORT command. The syntax of the command is:

PORT xx,xx,xx,xx,yy,yy

where xx,xx,xx,xx represents the 32-bit client IP address and yy,yy represents the 16-bit port number, e.g. 192,168,1,1,0, 21 translates to 192.168.1.1 at 0 x 256 + 21 = port 21.

According to the RFC, the server always uses port 20 as the data connection port.[1] The RFC states, "The server-process default data port is the port adjacent to the control connection port (i.e., L-1)."

Likewise, in TCP/IP Illustrated, Volume 1, Richard Stevens states, "The server's end of the data connection always uses port 20."[2]

Because the FTP protocol allows the from-anywhere-to-anywhere connection, the "client" IP address stated in the PORT command can actually be any IP address.

For example, let's say a user wants to download a file from an FTP server on his network to which he is not allowed to connect. However, the user knows there is a world-writeable directory on a different FTP server, one which is trusted to connect to the restricted server. There are three pieces of this particular flavor of the bounce attack that happen locally on the attack machine. First, the attacker opens an FTP connection on his or her own machine to set up a passive listener, noting the address and port returned from the PASV command. Second, the attacker creates an empty file and stores it in the same directory from which the listener is initiated. Third, s/he creates a file containing FTP

[1] Postel, J., Reynolds, J., "File Transfer Protocol," RFC 959, ISI, October 1985
URL: http://www.ietf.org/rfc/rfc959.txt (16 June 2002)
[2] TCP/IP Illustrated, Volume1 by W. Richard Stevens, Chapter 27, "FTP Protocol," page 425

logon and file retrieval commands, including the PORT command constructed with the port information noted in the first step and directing the data to be written to the empty file on the attacker's machine.

To exploit the FTP port command weakness, the client connects to the FTP server, uploads a script which contains a PORT command to his own machine, then issues a PORT command to a restricted third machine and launches the script which retrieves restricted data back to his own machine.

There are two victims in this scenario. Victim one is, of course, the restricted machine, which was duped into providing data to the attacker. Victim two is the bounce point, which provided the connection to the restricted machine.

This is just one example of the use of the FTP Bounce attack that involves the three key elements of the attack - the data and control connections on ports 20 and 21, respectively, and the use of a PORT command which references neither the client nor server involved in the initial open connections.

The Test Bed

In my test lab, I used Internet Security System's RealSecure as the IDS running on a Windows 2000 Advanced Server. Evaluation software is available at http://www.iss.net. My victim is a Windows 2000 Advanced Server running ISS RealSecure Server Sensor v.6.5.2002. My attackers are a Windows 2000 Professional workstation and a Linux RedHat 7.2 workstation.

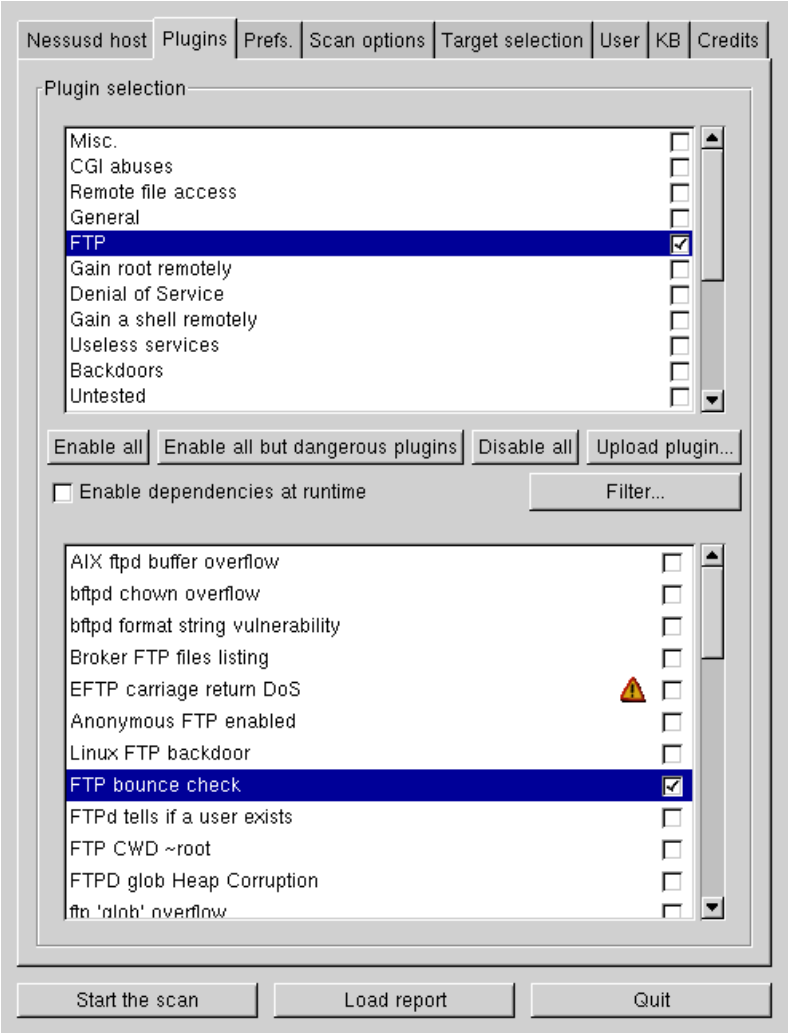
Let me reiterate that these are comparisons of attack simulation programs against an actual attack. The specifics of the IDS in place are not my focal point and, therefore, I will not discuss the architecture, signature set, whether one IDS is better than another, etc. The important piece of information relevant to the IDS itself is that the preceding attack triggered FTP Bounce alerts. The question was how the attack simulators would fare in the same arena.

Each attack method was executed 4 times in 10-minute intervals to test the consistency of the results. Using the actual attack discussed above, the IDS triggered every time. Thus, the litmus is set.

The Nessus Test

First up is a favorite among Linux users, the freeware vulnerability assessment program, Nessus (available at http://www.nessus.org). Nessus runs on *nix platforms and will scan any operating system hosting the appropriate client for

vulnerabilities chosen from a pre-defined database of exploits. To run an exploit simulation, one needs only to specify the target, select the desired exploit(s) from a checklist and hit "Start the scan."



The following tcpdump capture illustrates Nessus' attack simulation at the packet level. The three-way handshake is initiated by "linuxattacker":

```
13:56:12.623313 linuxattacker.32826 > victim.21: S
1290780899:1290780899(0) win 5840 <mss 1460,sackOK,timestamp 44182295
0,nop,wscale 0> (DF)
13:56:12.623471 victim.21 > linuxattacker.32826: S
3168123705:3168123705(0) ack 1290780900 win 17520 <mss
1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sackOK> (DF)
13:56:12.623626 linuxattacker.32826 > victim.21: . ack 1 win 5840
<nop,nop,timestamp 44182295 0> (DF)
```

Nessus then logs into the FTP service, in this case logging in as "administrator":

```
13:56:12.624987 linuxattacker.32826 > victim.21: P 1:21(20) ack 51
```

```
win 5840 <nop,nop,timestamp 44182295 28462632> (DF)
0x0000       4500 0048 ebf7 4000 4006 b113 0a1a 44c7 E..H..@.@.....D.
0x0010       0a1a 44aa 803a 0015 4cef c0e4 bcd5 bb6c ..D..:..L......l
0x0020       8018 16d0 f979 0000 0101 080a 02a2 2b17 .....y........+.
0x0030       01b2 4e28 5553 4552 2061 646d 696e 6973 ..N(USER.adminis
0x0040       7472 6174 6f72 0d0a                     trator..
13:56:12.625265 victim.21 > linuxattacker.32826: P 51:93(42) ack 21
win 17500 <nop,nop,timestamp 28462632 44182295> (DF)
0x0000       4500 005e 2e6d 4000 8006 2e88 0a1a 44aa E..^.m@.......D.
0x0010       0a1a 44c7 0015 803a bcd5 bb6c 4cef c0f8 ..D....:...lL...
0x0020       8018 445c 3d69 0000 0101 080a 01b2 4e28 ..D\=i........N(
0x0030       02a2 2b17 3333 3120 5061 7373 776f 7264 ..+.331.Password
0x0040       2072 6571 7569 7265 6420 666f 7220 6164 .required.for.ad
0x0050       6d69 6e69 7374 7261 746f 722e 0d0a      ministrator...
13:56:12.625603 linuxattacker.32826 > victim.21: P 21:36(15) ack 93
win 5840 <nop,nop,timestamp 44182295 28462632> (DF)
0x0000       4500 0043 ebf8 4000 4006 b117 0a1a 44c7 E..C..@.@.....D.
0x0010       0a1a 44aa 803a 0015 4cef c0f8 bcd5 bb96 ..D..:..L.......
0x0020       8018 16d0 4e04 0000 0101 080a 02a2 2b17 ....N.........+.
0x0030       01b2 4e28 5041 5353 2050 6173 7377 6f72 ..N(PASS.passwor
0x0040       640d 0a                                 d..
13:56:12.628815 victim.21 > linuxattacker.32826: P 93:128(35) ack 36
win 17485 <nop,nop,timestamp 28462632 44182295> (DF)
0x0000       4500 0057 2e6e 4000 8006 2e8e 0a1a 44aa E..W.n@.......D.
0x0010       0a1a 44c7 0015 803a bcd5 bb96 4cef c107 ..D....:....L...
0x0020       8018 444d 5942 0000 0101 080a 01b2 4e28 ..DMYB........N(
0x0030       02a2 2b17 3233 3020 5573 6572 2061 646d ..+.230.User.adm
0x0040       696e 6973 7472 6174 6f72 206c 6f67 6765 inistrator.logge
0x0050       6420 696e 2e0d 0a                       d.in...
```

Then Nessus issues a PORT command, referencing an arbitrary machine IP address:

```
13:56:12.631767 linuxattacker.32826 > victim.21: P 36:59(23) ack 128
win 5840 <nop,nop,timestamp 44182296 28462632> (DF)
0x0000       4500 004b ebf9 4000 4006 b10e 0a1a 44c7 E..K..@.@.....D.
0x0010       0a1a 44aa 803a 0015 4cef c107 bcd5 bbb9 ..D..:..L.......
0x0020       8018 16d0 e564 0000 0101 080a 02a2 2b18 .....d........+.
0x0030       01b2 4e28 504f 5254 2031 302c 3236 2c36 ..N(PORT.10,26,6
0x0040       382c 3137 312c 302c 3231 0a             8,171,0,21.
```

And that's it.

In the next frame, Nessus shuts down the FTP connection:

```
13:56:12.632506 linuxattacker.32826 > victim.21: F 59:59(0) ack 155
win 5840 <nop,nop,timestamp 44182296 28462632> (DF)
13:56:12.632660 victim.21 > linuxattacker.32826: . ack 60 win 17462
<nop,nop,timestamp 28462632 44182296> (DF)
13:56:12.632698 linuxattacker.32826 > victim.21: R 60:60(0) ack 155
win 5840 <nop,nop,timestamp 44182296 28462632> (DF)
13:56:12.632803 linuxattacker.32826 > victim.21: R
1290780959:1290780959(0) win 0 (DF)
```

Nessus did not open a data connection, only the control connection from which the PORT command was issued.

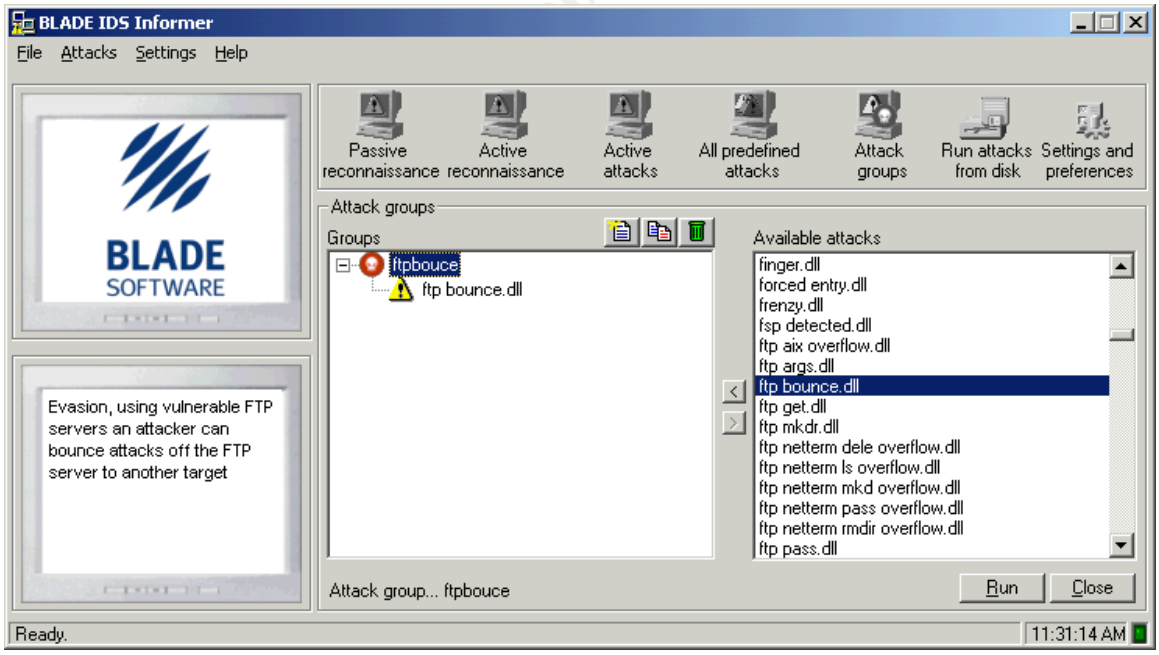
On my test IDS, this attack alerted 2 out of 4 times. Your mileage may vary, depending on your IDS. The Nessus behavior would likely trigger a signature

looking exclusively for PORT usage or arbitrary port addressing in the PORT command. However, it may be missed by a signature looking for port 20 activity in addition to the PORT command.

The IDS Informer Test

Next up is an up-and-comer on the Windows platform, Blade Software's IDS Informer. A perusal of the SecurityFocus focus-IDS mailing list archives indicates that Blade's software is growing in popularity among security consultants. Informer is a commercial product whose sole mission in life is to test IDS signatures. Evaluation software is available for download from the company's website, http://www.gui2000.com. A 7-day evaluation license is issued upon notification to the company of a customer ID which gets generated when Informer is installed and run the first time. The attack database is updated on a periodic basis by downloading "Attack Packs." Informer contains attack simulations for Windows, *nix, and Cisco vulnerabilities and exploits.

Setting up the FTP Bounce attack simulation is point-and-click, much like Nessus.



Once the desired attack (or attacks) have been selected, clicking "Run" starts the simulation.

Here's how it looked at the packet level. The connection from the attack machine, "winattacker," to the victim forks with a Reset at the third step of the 3-

way handshake into 2 distinct connections, each machine communicating from its own port 21 to the other's port 3161:

```
15:05:18.011750 winattacker.3161 > victim.21: S
1340719509:1340719509(0) win 32120 <mss 1460,sackOK,timestamp
233206378 0,nop,wscale 0> (DF)
15:05:18.011909 victim.21 > winattacker.3161: S
4052359975:4052359975(0) ack 1340719510 win 17520 <mss
1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sackOK> (DF)
15:05:18.012101 winattacker.3161 > victim.21: R
1340719510:1340719510(0) win 0
15:05:18.042560 winattacker.21 > victim.3161: S
503322329:503322329(0) ack 1340719510 win 10136 <nop,nop,timestamp
34757351 233206378,nop,wscale 0,mss 1460> (DF)
15:05:18.042672 winattacker.3161 > victim.21: . ack 745929651 win
32120 <nop,nop,timestamp 233206378 34757351> (DF)
```

The second connection, from the attacker port 21 to the victim port 3161, presents itself as a different, separate machine, the bounce point. Keep in mind that in this test, both machines are Windows 2000:

```
15:05:18.042714 winattacker.21 > victim.3161: P 1:43(42) ack 1 win
10136 <nop,nop,timestamp 34757353 233206378> (DF)
0x0000       4500 005e 1901 4000 0306 c139 0a1a 4482 E..^..@....9..D.
0x0010       0a1a 44aa 0015 0c59 1e00 16da 4fe9 c196 ..D....Y....O...
0x0020       8018 2798 4b4d 0000 0101 080a 0212 5ae9 ..'.KM........Z.
0x0030       0de6 726a 3232 3020 6469 6162 6c6f 2046 ..rj220.diablo.F
0x0040       5450 2073 6572 7665 7220 2853 756e 4f53 TP.server.(SunOS
0x0050       2035 2e36 2920 7265 6164 792e 0d0a      .5.6).ready...
```

The first PORT command is issued, from the first connection from the attacker's port 3161 to the victim's port 21:

```
15:05:18.057586 winattacker.3161 > victim.21: P 47:70(23) ack
745929786 win 32120 <nop,nop,timestamp 233208835 34758937> (DF)
0x0000       4500 004b ae34 4000 0306 2c19 0a1a 4482 E..K.4@...,...D.
0x0010       0a1a 44aa 0c59 0015 4fe9 c1c4 1e00 1761 ..D..Y..O......a
0x0020       8018 7d78 7ba1 0000 0101 080a 0de6 7c03 ..}x{.........|.
0x0030       0212 6119 706f 7274 2031 3732 2c31 362c ..a.port.172,16,
0x0040       302c 3332 2c31 322c 3732 0a             0,32,12,72.
```

Which is answered by the second connection:

```
15:05:18.058823 winattacker.21 > victim.3161: P 136:166(30) ack 70
win 10136 <nop,nop,timestamp 34759809 233208835> (DF)
0x0000       4500 0052 1906 4000 0306 c140 0a1a 4482 E..R..@....@..D.
0x0010       0a1a 44aa 0015 0c59 1e00 1761 4fe9 c1db ..D....Y...aO...
0x0020       8018 2798 b0a6 0000 0101 080a 0212 6481 ..'...........d.
0x0030       0de6 7c03 3230 3020 504f 5254 2063 6f6d ..|.200.PORT.com
0x0040       6d61 6e64 2073 7563 6365 7373 6675 6c2e mand.successful.
```

It is on this connection that the bounce occurs:

```
15:05:18.063925 winattacker.21 > victim.3161: P 166:235(69) ack 75
win 10136 <nop,nop,timestamp 34760012 233209038> (DF)
0x0000       4500 0079 1907 4000 0306 c118 0a1a 4482 E..y..@.......D.
0x0010       0a1a 44aa 0015 0c59 1e00 177f 4fe9 c1e0 ..D....Y....O...
```

```
0x0020        8018 2798 8b9f 0000 0101 080a 0212 654c ..'...........eL
0x0030        0de6 7cce 3135 3020 4153 4349 4920 6461 ..|.150.ASCII.da
0x0040        7461 2063 6f6e 6e65 6374 696f 6e20 666f ta.connection.fo
0x0050        7220 2f62 696e 2f6c 7320 2831 3732 2e31 r./bin/ls.(172.1
0x0060        362e 302e 3332 2c33 3134 3429 2028 3020 6.0.32,3144).(0.
0x0070        6279 7465 7329 2e0d 0a                  bytes)...
```

Then the data connection opens and "transfers" 264 bytes of garbage data:

```
15:05:18.079124 winattacker.22 > victim.3154: P 1448:1712(264) ack 1
win 10136 <nop,nop,timestamp 34760014 233204253> (DF) [tos 0x10]
```

All connections close upon completion of the "data transfer."

As you can see, Informer provides an interesting method of simulating a third-party connection, utilizing the attack machine as a stand-in. The attack elements are all there although the data connection is opened on port 22, the SSH port, as opposed to the more commonly recognized port 20, as stated in the RFC. In this test, Informer triggered the IDS 3 out of 4 times.

Conclusion

Nothing in the test environment changed between each attack. Accounting for the unsuccessful responses, also known as false negatives, would require closer examination of the attack simulation software and the test environment itself. And the question is should the IDS be tuned to respond to the simulation software? An analyst relying solely on simulation software for signature testing may inadvertently do just that. Remember, the actual attack triggered 100 percent of the time.

A point-and-click solution to IDS signature testing is a well-intentioned idea but not without two significant traps.

First, at this point in the maturity level of currently available software, a reliable second methodology is necessary to ensure consistent results. As the test above indicates, the most reliable test method is the actual attack. Duplicating the attack with a simulation requires an increase in time and effort, which could be better spent tuning signatures and catching bad guys.

Second, an off-the-shelf software product may lead the inexperienced to believe that expertise is unnecessary. Over-burdened system administrators who are expected to provide quality security solutions would - and probably do - jump at a quick and easy point-and-click alternative to learning how to truly identify the tools and techniques used against Internet-connected systems.

Both traps are sidestepped by experience. There simply is no replacement for

practical knowledge.

References

Postel, J., Reynolds, J., "File Transfer Protocol," RFC 959, ISI, October 1985
URL: http://www.ietf.org/rfc/rfc959.txt (16 June 2002)

Stevens, W. Richard, "TCP/IP Illustrated, Volume 1, The Protocols," Addison-Wesley Publishing, 1994

Hobbit, "The FTP Bounce Attack," posted to bugtraq July 12, 1995
URL: http://www.insecure.org/nmap/hobbit.ftpbounce.txt (14 June 2002)

Carnegie Mellon University, CERT Coordination Center, CERT Advisory CA-1997-27 FTP Bounce, December 10, 1997, rev. April 3, 2002
URL: http://www.cert.org/advisories/CA-1997-27.html (14 June 2002)

Carnegie Mellon University, CERT Coordination Center, "Problems With The FTP PORT Command or Why You Don't Want Just Any PORT in a Storm," 1998
URL: http://www.cert.org/tech_tips/ftp_port_attacks.html (14 June 2002)

Farmer, Dan, Venema, Wietse, "Improving The Security of Your Site By Breaking Into It," date unknown
URL: http://www.fish.com/security/admin-guide-to-cracking.html (16 June 2002)

Assignment #2 - Network Detects

Introduction

Detects 1 and 2 were captured on my home network, collected on a hardened Windows 2000 Professional machine. In place of a "traditional" intrusion detection system, I logged the alerts generated by ZoneAlarm Pro v.3 ("ZA") personal firewall on the collection host. These events were correlated to logs generated by windump (the win32 port of tcpdump), version 3.6.2, on the collection host and Snort, version 1.8.6 (win32) on a second Windows 2000 Professional host. Snort alerting is not available under win32, which is why I used ZoneAlarm as the primary alerting program.

The ZA alerts conform to the following format:

Event type,date,timestamp(local and GMT variance),source IP address:source port,destination IP address:destination port,protocol(flags:<flag>)

For example:

```
FWIN,2002/06/18,22:36:46 -7:00
GMT,66.86.14.167:1173,192.168.1.1:139,TCP(flags:S)
```

The event type "FWIN" translates to the firewall blocking an inbound TCP SYN packet. In the above example, the alert was received on June 18, 2002 at 10:36:46 p.m. local time, 7 hours less than Greenwich Mean Time, originating from 66.86.14.167 from its port 1173, directed toward port 139 to 192.168.1.1.

The windump and Snort logs follow their respective well-known formats, discussed in detail in Intrusion Signatures and Analysis, Chapter 1[3].

Detect #1 - LPRng Scan

Event Traces

The following alert was reported by ZA:

```
FWIN,2002/06/20,04:36:50 -7:00
GMT,211.197.180.9:1661,xxx.yyy.1.103:515,TCP (flags:S)
```

Internal correlating data was captured from the router by Snort. Note the "@in" IP address:

---

[3] Northcutt, Stephen; Cooper, Mark; Fearnow, Matt; Frederick, Karen, Intrusion Signatures and Analysis. New Riders Publishing, 2001.

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

06/20-04:36:51.398326 0:4:5A:F7:68:18 -> FF:FF:FF:FF:FF:FF type:0x800
len:0x9C
xxx.yyy.zzz.1:2651 -> xxx.yyy.zzz.255:162 UDP TTL:150 TOS:0x0 ID:0
IpLen:20 DgmLen:142
Len: 122
30 82 00 6E 02 01 00 04 06 70 75 62 6C 69 63 A4  0..n.....public.
82 00 5F 06 0A 2B 06 01 04 01 98 15 02 02 01 40  .._..+.........@
04 C0 A8 01 01 02 01 06 02 01 01 43 04 04 C9 19  ...........C....
67 30 82 00 3D 30 82 00 39 06 0A 2B 06 01 04 01  g0..=0..9..+....
98 15 01 01 00 04 82 00 29 40 69 6E 20 32 31 31  ........)@in 211
2E 31 39 37 2E 31 38 30 2E 39 20 31 36 36 31 20  .197.180.9 1661
xx xx xx 2E yy yy yy 2E 31 2E 31 30 33 20 35 31  xxx.yyy.1.103 51
35 0A                                            5.

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

Source of Trace

This trace was captured on my home network, consisting of a single Class C subnet.

Detect Generated By

ZoneAlarm Pro and Snort running on two different machines.

Probability The Source Address Was Spoofed

The probability is low. Host scans are typical of reconnaissance efforts. The would-be attacker was scanning for a particular service which means he or she was looking for a particular vulnerability. Vulnerable machines revealed by the scan undoubtedly report to the attacker. The attacker would need to receive those reports in order to proceed with the actual attack. Additionally, there were no further scans of this nature received during the collection period of a week which would indicate that the attacker made no attempt to obfuscate their scan by using more than one host to create a lot of traffic "noise." The attack host is likely either the attacker's personal machine or a previously compromised machine s/he "owns," i.e., a machine to which s/he has gained access.

Description of the Attack

The attacker scanned for the LPRng input validation vulnerability. LPRng is a print service management package included in Linux and BSD distributions which runs on TCP port 515. In version 3.6.24 and earlier, the code is missing format string arguments of "%s" in both calls to syslog() in the following section (identified by Chris Evans[4]):

```
LPRng-3.6.24/src/common/errormsg.c, use_syslog()
---
static void use_syslog(int kind, char *msg)
[...]
# ifdef HAVE_OPENLOG
        /* use the openlog facility */
        openlog(Name, LOG_PID | LOG_NOWAIT, SYSLOG_FACILITY );
        syslog(kind, msg);
        closelog();

# else
    (void) syslog(SYSLOG_FACILITY | kind, msg);
# endif                                                    /*
HAVE_OPENLOG */
[...]
```

The missing arguments may allow user-supplied input as format string parameters to susceptible *snprintf() calls to overwrite arbitrary addresses in the printing service's address space. A remote user could potentially exploit this vulnerability to corrupt the printer daemon's execution and gain root access to the machine.

The LPRng vulnerability is described in detail in the following advisories:

- CERT Advisory CA-2000-22
- CVE-2000-0917
- CIAC Bulletin L-004

Attack Mechanism

The method of attack in this case is most likely a scan across one or more subnets, looking for hosts listening on TCP port 515. The LPRng vulnerability is widely known and exploit scripts are relatively easy to find on the Internet. An attacker would first need to identify potential victims before attempting to run an LPRng exploit. Scanning for hosts with port 515 open is the "shotgun" approach to reconnaissance, to narrow down the field to specific, vulnerable targets.

There also exists the possibility that this scan was the result of a worm,

[4] Evans, Chris, "Format Strings: bug #2 LPRng." Bugtraq post. 26 Sep 2000. URL: http://online.securityfocus.com/archive/1/85002 (8 Sep. 2002).

lpdw0rm. This worm gains entry by exploiting the LPRng vulnerability and plants two backdoor access methods in victim hosts then moves on to find other victims. The worm's code includes the script "scan.sh," which provides three functions. First, it runs "randb," which randomly generates the first two octets of a target IP address range, avoiding those earmarked for internal addressing (0 through 10, 49, any number over 230 and the first two octets of 192.168). Next, it runs "pscan," a port scanner, which searches the randomly generated address space. Finally, it runs ".hack," which launches the LPRng exploit.

Network address translation is performed at the router so the latter possibility is viable, in spite of the fact that the trace identified the targeted host using the internal private network address.

Correlations

This attack attempt was identified on my network by both the host-based firewall and Snort running on an internal host. Each capture is noted under "Event Traces," above.

In the wild, similar activity was reported on Incidents.org's Intrusions List on June 17, 2002, by Ken Connelly, Systems and Operations Manager, ITS - Network Services at the University of Northern Iowa in Cedar Falls, Iowa (relevant trace only)[5]:

```
The following extracts show the beginning and ending of scan activity
was [sic] detected on my network. The number following each set is
the total number of probes for that source. Timestamps are GMT-0500.

<snip>
Jun 14 02:28:53 200.72.14.66:4497 -> xxx.yyy.0.37:515 SYN ******S*
Jun 14 02:28:53 200.72.14.66:4479 -> xxx.yyy.0.19:515 SYN ******S*
Jun 14 02:28:53 200.72.14.66:4475 -> xxx.yyy.0.15:515 SYN ******S*
Jun 14 02:28:53 200.72.14.66:4482 -> xxx.yyy.0.22:515 SYN ******S*
Jun 14 02:28:53 200.72.14.66:4486 -> xxx.yyy.0.26:515 SYN ******S*
Jun 14 02:28:53 200.72.14.66:4488 -> xxx.yyy.0.28:515 SYN ******S*
Jun 14 02:28:53 200.72.14.66:4506 -> xxx.yyy.0.46:515 SYN ******S*
Jun 14 02:28:53 200.72.14.66:4494 -> xxx.yyy.0.34:515 SYN ******S*
[...]
<snip>
```

While the date of the scan captured by Ken Connelly, June 14, preceded the scan against my network by several days, it shows the interest in port 515 was on the rise.

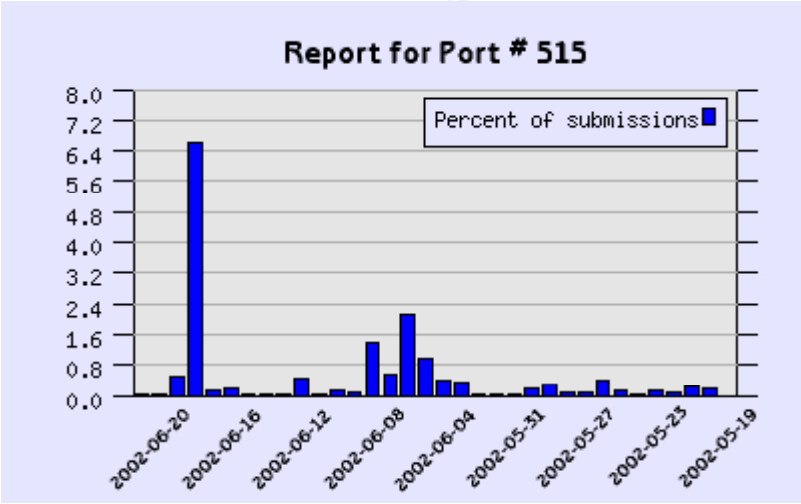
Likewise, Laurie Zirkle, from Communications Network Services at Virginia

[5] Connelly, Ken, "[LOGS] Summary of Large Scale Portscanning Detects," 17 June 2002. URL: http://cert.uni-stuttgart.de/archive/intrusions/2002/06/msg00155.html (7 September 2002).

Tech, posted the following scan[6], which occurred several days later:

```
Jun 23 15:36:33 142.132.195.2:1545 -> a.b.w.51:515 SYN ******S*
Jun 23 15:36:33 142.132.195.2:1615 -> a.b.w.62:515 SYN ******S*
Jun 23 15:36:35 142.132.195.2:1444 -> a.b.w.33:515 SYN ******S*
Jun 23 15:36:35 142.132.195.2:1499 -> a.b.w.37:515 SYN ******S*
Jun 23 15:36:35 142.132.195.2:1500 -> a.b.w.38:515 SYN ******S*
Jun 23 15:36:36 142.132.195.2:1547 -> a.b.w.53:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3427 -> a.b.w.68:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3428 -> a.b.w.69:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3445 -> a.b.w.86:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3460 -> a.b.w.101:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3473 -> a.b.w.114:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3474 -> a.b.w.115:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3518 -> a.b.w.159:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3519 -> a.b.w.160:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3527 -> a.b.w.168:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3554 -> a.b.w.195:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3594 -> a.b.w.235:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3596 -> a.b.w.237:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3598 -> a.b.w.239:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3599 -> a.b.w.240:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3606 -> a.b.w.247:515 SYN ******S*
Jun 23 15:36:38 142.132.195.2:3608 -> a.b.w.249:515 SYN ******S*
Jun 23 15:36:41 142.132.195.2:3557 -> a.b.w.198:515 SYN ******S*
```

On June 20, the date of the attack attempt on my host, the Distributed Intrusion Detection System at DShield.org reported 247 port 515 connection attempts in the wild. There was a significant spike of activity on this port in the days immediately preceding the attempt, as noted in DShield's Port Report graph below:



DShield also provides an IP address lookup to find out whether a particular IP has been associated with port-specific scans or attack attempts. In this case, the attacker's IP had not been previously reported.

[6] Zirkle, Laurie, scan logs, June 2002
URL: http://rdweb.cns.vt.edu/~lat/log_archives/020623.txt (7 September 2002)

A "whois" search of the IP address on ARIN led me to APNIC, which, in turn, referred me to KRNIC, the Korea Internet Information Service. The IP address query finally returned the information that the IP is owned by a Korean Internet Service Provider, KORNET:

```
query: 211.197.180.9

# ENGLISH

KRNIC is not ISP but National Internet Registry similar with APNIC.
The IP address is allocated and still held by the following ISP, or
they did not update whois information after assigning to end-user.

Please see the following ISP contacts for relevant information
or network abuse complaints.

[ ISP Organization Information ]
Org Name      : Korea Telecom
Service Name  : KORNET
Org Address   : 206 Jungja-dong, Bundang-gu, Sungnam city, Gyunggi-
do, Korea, 463-711

[ ISP IP Admin Contact Information ]
Name          : Lee Heung-Gu
Phone         : +82-2-747-9213
Fax           : +82-2-747-8701
E-Mail        : ip@ns.kornet.net

[ ISP IP Tech Contact Information ]
Name          : Kang Won
Phone         : +82-2-747-9213
Fax           : +82-2-747-8701
E-mail        : ip@ns.kornet.net

[ ISP Network Abuse Contact Information ]
Name          : Kim Jin-Won
Phone         : +82-2-3675-1499
Fax           : +82-2-747-8701
E-mail        : abuse@kornet.net
```

A search of Usenet, now available as "Google Groups," revealed that KORNET is a source of a great deal of spam. However, there are no references to this particular IP address, nor an attack of this or similar natures attributed to this IP address or the ISP.

Evidence of Active Targeting

Active targeting is highly unlikely. The target is a Windows-based host. The scanned service and vulnerability do not exist on any machines on my network. My external IP address was likely within a range of scanned addresses. This is further evidenced by the fact that I received only one scan for port 515. Had the attacker been actively targeting my system, I would have expected to see

several scan attempts, whether from different hosts scanning for a single port or a full service port scan originating from this same IP address.

Severity

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

Criticality = 1
The scanned system is a Windows-based client workstation with the default admin shares (admin$, C$, IPC$) disabled. Privilege escalation is still possible, using a buffer overflow method, but not likely unless the system or network had been actively targeted.

Lethality = 5
Had this attack been successful, the attacker could potentially have gained administrative privileges to this system and to my network.

System Countermeasures = 5
The system scanned is fully patched, firewalled, and monitored by Snort.

Network Countermeasures = 1
The perimeter firewall was disabled for this exercise. The router performs network address translation, which can obfuscate the internal network, but the router itself was vulnerable. Snort monitors the internal network, however, because it ran on a Windows machine for this exercise, the alerting feature was not available.

Therefore, (1 + 5) - (5 + 1) = Severity of 0

While the network defenses were weak, the host itself was ostensibly invulnerable by nature of the OS and limited services available, further protected by the firewall, thus the attack attempt failed.

Defensive Recommendations

The router performs network address translation and each internal host is firewalled and monitored. All systems, both Red Hat and Windows, are patched to current levels. The targeted service, LPRng, is not running on any of my systems. Snort is running on two or more hosts at any given time, monitoring the internal network. Although the attack failed, it is still advisable to firewall the router to provide a perimeter defense.

Multiple Choice Test Question

The following operating systems are immune to the LPRng attack.

a) Linux Red Hat 7.x
b) FreeBSD
c) Windows 2000
d) Solaris 2.x

Answer: c
While Windows NT and 2000 provide heterogenous print daemon support and are vulnerable to denial of service attacks on the LPR and LPD ports, Windows is not vulnerable to this specific exploit because the vulnerability is based on a weakness in the service management package code, written for *nix platforms.

Detect #2 - SQLSpida.B Worm

Event Traces

The following alerts were reported by ZA over a three-day period, from the evening of June 18 through June 21, 2002:

```
FWIN,2002/06/19,01:25:58 -7:00
GMT,211.57.229.135:2795,xxx.yyy.1.103:1433,TCP (flags:S)
FWIN,2002/06/19,02:36:02 -7:00
GMT,62.30.230.76:3600,xxx.yyy.1.103:1433,TCP (flags:S)
FWIN,2002/06/19,04:33:04 -7:00
GMT,61.185.243.28:1508,xxx.yyy.1.103:1433,TCP (flags:S)

FWIN,2002/06/20,01:40:08 -7:00
GMT,24.190.6.133:1743,xxx.yyy.1.103:1433,TCP (flags:S)

FWIN,2002/06/20,22:29:44 -7:00
GMT,211.225.96.215:4091,xxx.yyy.1.103:1433,TCP (flags:S)
FWIN,2002/06/20,23:12:18 -7:00
GMT,61.255.71.181:4496,xxx.yyy.1.103:1433,TCP (flags:S)
FWIN,2002/06/21,02:21:30 -7:00
GMT,211.44.159.222:3053,xxx.yyy.1.103:1433,TCP (flags:S)
FWIN,2002/06/21,05:48:06 -7:00
GMT,61.82.129.247:1666,xxx.yyy.1.103:1433,TCP (flags:S)
```

Internal correlating data was captured on the router by windump:

```
01:25:58.480696 xxx.yyy.zzz.1.2550 > xxx.yyy.zzz.255.162:  Trap(97)
E:3093.2.2.1 xxx.yyy.zzz.1 enterpriseSpecific[specific-trap(1)!=0]
70502747 E:3093.1.1.0="@in 211.57.229.135 2795 xxx.yyy.1.103 1433^J"
0x0000       4500 0090 0000 0000 9611 a00c c0a8 0101E...............
```

```
0x0010       c0a8 01ff 09f6 00a2 007c 11b9 3082 0070 .........|..0..p
0x0020       0201 0004 0670 7562 6c69 63a4 8200 6106 .....public...a.
0x0030       0a2b 0601 0401 9815 0202 0140 04c0 a801 .+.........@....
0x0040       0102 0106 0201 0143 0404 33c9 5b30 8200 .......C..3.[0..
0x0050       3f30 8200 3b06 0a2b 0601 0401 9815 0101 ?0..;..+........
0x0060       0004 8200 2b40 696e 2032 3131 2e35 372e ....+@in.211.57.
0x0070       3232 392e 3133 3520 3237 3935 20xx xxxx 229.135.2795.xxx
0x0080       2eyy yyyy 2e31 2e31 3033 2031 3433 330a .yyy.1.103.1433.
02:36:03.665610 xxx.yyy.zzz.1.2554 > xxx.yyy.zzz.255.162:  Trap(95)
E:3093.2.2.1 xxx.yyy.zzz.1 enterpriseSpecific[specific-trap(1)!=0]
70923290 E:3093.1.1.0="@in 62.30.230.76 3600 xxx.yyy.1.103 1433^J"
0x0000       4500 008e 0000 0000 9611 a00e c0a8 0101 E...............
0x0010       c0a8 01ff 09fa 00a2 007a 9982 3082 006e .........z..0..n
0x0020       0201 0004 0670 7562 6c69 63a4 8200 5f06 .....public..._.
0x0030       0a2b 0601 0401 9815 0202 0140 04c0 a801 .+.........@....
0x0040       0102 0106 0201 0143 0404 3a34 1a30 8200 .......C..:4.0..
0x0050       3d30 8200 3906 0a2b 0601 0401 9815 0101 =0..9..+........
0x0060       0004 8200 2940 696e 2036 322e 3330 2e32 ....)@in.62.30.2
0x0070       3330 2e37 3620 3336 3030 20xx xxxx 2eyy 30.76.3600.xxx.y
0x0080       yyyy 2e31 2e31 3033 2031 3433 330a      yyy.1.103.1433.

04:33:05.734036 xxx.yyy.zzz.1.2558 > xxx.yyy.zzz.255.162:  Trap(96)
E:3093.2.2.1 xxx.yyy.zzz.1 enterpriseSpecific[specific-trap(1)!=0]
71625538 E:3093.1.1.0="@in 61.185.243.28 1508 xxx.yyy.1.103 1433^J"
0x0000       4500 008f 0000 0000 9611 a00d c0a8 0101 E...............
0x0010       c0a8 01ff 09fe 00a2 007b 35b1 3082 006f .........{5.0..o
0x0020       0201 0004 0670 7562 6c69 63a4 8200 6006 .....public...`.
0x0030       0a2b 0601 0401 9815 0202 0140 04c0 a801 .+.........@....
0x0040       0102 0106 0201 0143 0404 44eb 4230 8200 .......C..D.B0..
0x0050       3e30 8200 3a06 0a2b 0601 0401 9815 0101 >0..:..+........
0x0060       0004 8200 2a40 696e 2036 312e 3138 352e ....*@in.61.185.
0x0070       3234 332e 3238 2031 3530 3820 xxxx xx2e 243.28.1508.192.
0x0080       yyyy yy2e 312e 3130 3320 3134 3333 0a   168.1.103.1433.

22:29:44.302409 xxx.yyy.zzz.1.2804 > xxx.yyy.zzz.255.162:  Trap(97)
E:3093.2.2.1 xxx.yyy.zzz.1 enterpriseSpecific[specific-trap(1)!=0]
86725422 E:3093.1.1.0="@in 211.225.96.215 4091 xxx.yyy.1.103 1433^J"
0x0000       4500 0090 0000 0000 9611 a00c c0a8 0101 E...............
0x0010       c0a8 01ff 0af4 00a2 007c 4f32 3082 0070 .........|O20..p
0x0020       0201 0004 0670 7562 6c69 63a4 8200 6106 .....public...a.
0x0030       0a2b 0601 0401 9815 0202 0140 04c0 a801 .+.........@....
0x0040       0102 0106 0201 0143 0405 2b53 2e30 8200 .......C..+S.0..
0x0050       3f30 8200 3b06 0a2b 0601 0401 9815 0101 ?0..;..+........
0x0060       0004 8200 2b40 696e 2032 3131 2e32 3235 ....+@in.211.225
0x0070       2e39 362e 3231 3520 3430 3931 20xx xxxx .96.215.4091.xxx
0x0080       2eyy yyyy 2e31 2e31 3033 2031 3433 330a .yyy.1.103.1433.

23:12:18.937773 xxx.yyy.zzz.1.2807 > xxx.yyy.zzz.255.162:  Trap(96)
E:3093.2.2.1 xxx.yyy.zzz.1 enterpriseSpecific[specific-trap(1)!=0]
86980936 E:3093.1.1.0="@in 61.255.71.181 4496 xxx.yyy.1.103 1433^J"
0x0000       4500 008f 0000 0000 9611 a00d c0a8 0101 E...............
0x0010       c0a8 01ff 0af7 00a2 007b 4067 3082 006f .........{@g0..o
0x0020       0201 0004 0670 7562 6c69 63a4 8200 6006 .....public...`.
0x0030       0a2b 0601 0401 9815 0202 0140 04c0 a801 .+.........@....
0x0040       0102 0106 0201 0143 0405 2f39 4830 8200 .......C../9H0..
0x0050       3e30 8200 3a06 0a2b 0601 0401 9815 0101 >0..:..+........
0x0060       0004 8200 2a40 696e 2036 312e 3235 352e ....*@in.61.255.
0x0070       3731 2e31 3831 2034 3439 3620 xxxx xx2e 71.181.4496.xxx.
0x0080       yyyy yy2e 312e 3130 3320 3134 3333 0a   yyy.1.103.1433.

02:21:31.647581 xxx.yyy.zzz.1.2810 > xxx.yyy.zzz.255.162:  Trap(97)
E:3093.2.2.1 xxx.yyy.zzz.1 enterpriseSpecific[specific-trap(1)!=0]
88116273 E:3093.1.1.0="@in 211.44.159.222 3053 xxx.yyy.1.103 1433^J"
0x0000       4500 0090 0000 0000 9611 a00c c0a8 0101 E...............
```

```
0x0010       c0a8 01ff 0afa 00a2 007c 36f9 3082 0070 .........|6.0..p
0x0020       0201 0004 0670 7562 6c69 63a4 8200 6106 .....public...a.
0x0030       0a2b 0601 0401 9815 0202 0140 04c0 a801 .+.........@....
0x0040       0102 0106 0201 0143 0405 408c 3130 8200 .......C..@.10..
0x0050       3f30 8200 3b06 0a2b 0601 0401 9815 0101 ?0..;..+........
0x0060       0004 8200 2b40 696e 2032 3131 2e34 342e ....+@in.211.44.
0x0070       3135 392e 3232 3220 3330 3533 20xx xxxx 159.222.3053.xxx
0x0080       2eyy yyyy 2e31 2e31 3033 2031 3433 330a .yyy.1.103.1433.
05:48:06.478098 xxx.yyy.zzz.1.2817 > xxx.yyy.zzz.255.162:  Trap(96)
E:3093.2.2.1 xxx.yyy.zzz.1 enterpriseSpecific[specific-trap(1)!=0]
89355829 E:3093.1.1.0="@in 61.82.129.247 1666 xxx.yyy.1.103 1433^J"
0x0000       4500 008f 0000 0000 9611 a00d c0a8 0101 E...............
0x0010       c0a8 01ff 0b01 00a2 007b 3717 3082 006f .........{7.0..o
0x0020       0201 0004 0670 7562 6c69 63a4 8200 6006 .....public...`.
0x0030       0a2b 0601 0401 9815 0202 0140 04c0 a801 .+.........@....
0x0040       0102 0106 0201 0143 0405 5376 3530 8200 .......C..Sv50..
0x0050       3e30 8200 3a06 0a2b 0601 0401 9815 0101 >0..:..+........
0x0060       0004 8200 2a40 696e 2036 312e 3832 2e31 ....*@in.61.82.1
0x0070       3239 2e32 3437 2031 3636 3620 xxxx xx2e 29.247.1666.192.
0x0080       yyyy yy2e 312e 3130 3320 3134 3333 0a   168.1.103.1433.
```

The alert generated at 1:40am on June 20 was captured on the router by Snort:

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

06/20-01:40:09.907964 0:4:5A:F7:68:18 -> FF:FF:FF:FF:FF:FF type:0x800
len:0x9C
xxx.yyy.zzz.1:2646 -> xxx.yyy.zzz.255:162 UDP TTL:150 TOS:0x0 ID:0
IpLen:20 DgmLen:142
Len: 122
30 82 00 6E 02 01 00 04 06 70 75 62 6C 69 63 A4  0..n.....public.
82 00 5F 06 0A 2B 06 01 04 01 98 15 02 02 01 40  .._..+.........@
04 C0 A8 01 01 02 01 06 02 01 01 43 04 04 B8 EB  ...........C....
F4 30 82 00 3D 30 82 00 39 06 0A 2B 06 01 04 01  .0..=0..9..+....
98 15 01 01 00 04 82 00 29 40 69 6E 20 32 34 2E  ........)@in 24.
31 39 30 2E 36 2E 31 33 33 20 31 37 34 33 20 xx  190.6.133 1743 x
xx xx 2E yy yy yy 2E 31 2E 31 30 33 20 31 34 33  xx.yyy.1.103 143
33 0A                                            3.

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

Source of Trace

This trace was captured on my home network, consisting of a single Class C subnet.

Detect Generated By

ZoneAlarm Pro, windump, and Snort running on different machines.

Probability The Source Address Was Spoofed

The probability is low. Worm activity moves from one active target to the next; the sources are likely previously infected hosts.

In the event these were individual scans capitalizing on the renewed interest in the SQL port 1433, host scans are typical of reconnaissance efforts. Vulnerable machines revealed by the scans undoubtedly report to the attacker(s). The attacker(s) would need to receive those reports in order to proceed with the actual attack. In this case, the attack hosts are likely either the attackers' personal machines or previously compromised machines that one or more attackers "owns," i.e., a machine to which s/he has gained access.

Description of the Attack

In May 2002, a new variant of an old vulnerability appeared in the wild, popularly known as the Spida Worm. The worm attempts to exploit the default Windows SQL 7 and SQL 2000 "SA" account null password setting. The worm seeks out machines left with the default of no password then leverages a feature in SQL to call functions outside of the database using extended stored procedures, which allows interaction with the operating system and/or network. The following files are enclosed in the worm's payload:

drivers\services.exe (contains Foundstone's fscan);
clemail.exe
pwdump2.exe
run.js
samdump.dll
sqldir.js
sqlexec.js
sqlinstall.bat
sqlprocess.js
timer.dll

The file "sqlexec.js" appears to be the heart of the exploit. It contains command-line functions called by "sqlinstall.bat" to test the connection by sending an echo, then enables the built-in guest account, sets a password, and adds the account to the Administrators and Domain Admins groups. Next, it connects to the victim as "guest." The worm then tests for cscript.exe, which is required for the code to run, and checks for previous infection. The files get copied to the victim then hidden, the guest account gets deactivated and removed from the admin groups, and a password is randomly generated for the SA account. This

ensures the worm writer alone will have "ownership" of the infected machine by cleaning up the vulnerability that let him/her in. This is a standard step in a system compromise.

System variables and network settings are collected through sqlprocess.js. This same script also calls pwdump2, which grabs passwords in conjunction with samdump.dll, and fscan, renamed as "services.exe," to scan for the next victim(s), looking for the SQL port of 1433, avoiding private network addresses and loopback by excluding the first octets of 10, 127, 172, and 192. Run.js triggers the worm when the next victim or victims are found.

The script also makes two changes to the registry key HKLM\System\CurrentControlSet\Services\NetDDE, under "ImagePath" and "Start," to ensure startup on reboot. Database table and row data is collected with sqldir.js. All collected output is directed to SEND.TXT, which clemail.exe sends to an email address in Singapore, ixltd@postone.com.

George Bakos, of Dartmouth College, posted a detailed code analysis to Incidents.org on May 21, 2002.[7]

Additionally, the following advisories discuss the Spida Worm:

- CERT Incident Note IN-2002-04
- CVE Candidate CAN-2000-1209
- Microsoft's Product Support Services Informational Alert on SQL Server

The original SA account default setting vulnerability is detailed in the CERT Vulnerability Note VU #635463.

Attack Mechanism

Given the timing, within a month of the first report of Spida, and the frequency of the attack attempts, this appears to be the work of the SQLSpida.B worm. It is possible that the scans resulted from renewed interest in SQL's SA account default, separate and apart from worm activity launched in May. However, take a look at the originating IP addresses:

211.57.229.135
62.30.230.76
61.185.243.28
24.190.6.133

[7] Bakos, George, "SQLsnake Code Analysis," 21 May 2002.
URL: http://www.incidents.org/diary/diary.php?id=157 (7 September 2002)

211.225.96.215
61.255.71.181
211.44.159.222
61.82.129.247

A pattern is clear. All but two addresses include a first octet of either 211 or 61.

Now let's look at the times associated with those six IP addresses:

211.57.229.135 - 2002/06/19,01:25:58
61.185.243.28 - 2002/06/19,04:33:04
211.225.96.215 - 2002/06/20,22:29:44
61.255.71.181 - 2002/06/20,23:12:18
211.44.159.222 - 2002/06/21,02:21:30
61.82.129.247 - 2002/06/21,05:48:06

Nothing especially impressive is indicated, until I ran DShield.org's IP Info Reports on the IP addresses. Here is the first IP of the 6. The administrator and technical contact names have been edited out to save space:

IP Address: 211.57.229.135
HostName: 211.57.229.135
DShield Profile: Country:
KR

Contact E-mail:
ip_AT_ns.pubnet.ne.kr (bounced)

Total Records against IP:
399

Number of targets:
374

Date Range:
2002-06-17 to 2002-06-17

Ports Attacked (up to 10):

Port	Attacks
1433	9

Fightback: sent to ip@ns.pubnet.ne.kr on 2002-06-04 02:46:59
no reply received

Whois: IP Address : 211.57.229.0-211.57.229.255
Connect ISP Name : PUBNET
Connect Date : 20000229
Registration Date : 20000310
Network Name : SEOCHO-ETH

[Organization Information]
Orgnization ID : ORG32586
Name : Seocho Electronic Technical High School
State : SEOUL
Address : 2727 Bangbae-dong Seocho-gu
Zip Code : 137-060

The second IP address:

IP Address: 61.185.243.28
HostName: www.xaeconomy.com.cn
DShield Profile: Country:

Contact E-mail:

Total Records against IP:
1096

Number of targets:
445

Date Range:
2002-06-11 to 2002-06-11

Ports Attacked (up to 10):

Port	Attacks
1433	382

Fightback: not sent

Whois: % Rights restricted by copyright. See
http://www.apnic.net/db/dbcopyright.html
% (whois7.apnic.net)

inetnum: 61.185.0.0 - 61.185.255.255
netname: CHINANET-SN
descr: CHINANET Shanxi(SN) province network
descr: Data Communication Division
descr: China Telecom
country: CN
admin-c: CH93-AP
tech-c: XC9-AP
mnt-by: MAINT-CHINANET
mnt-lower: MAINT-CHINANET-SHAANXI
changed: hostmaster@ns.chinanet.cn.net 20010216
source: APNIC

The third:

IP Address: 211.225.96.215
HostName: 211.225.96.215
DShield Profile: Country:
KR

Contact E-mail:
abuse_AT_kornet.net (bounced)

Total Records against IP:
4

Number of targets:
2

Date Range:
2002-06-25 to 2002-06-25

Ports Attacked (up to 10):

Port	Attacks
1433	1

Fightback: not sent

Whois: IP Address : 211.225.93.0-211.225.96.255
Connect ISP Name : KORNET
Connect Date : 20001001
Registration Date : 20001024
Network Name : KORNET-INFRA-JEONBUK

[Organization Information]
Orgnization ID : ORG130170
Name : JEONBUK
State : CHONBUK
Address : 1274-1 1KA DEOKJINDONG JEONJUSI
Zip Code : 561-190

The fourth:

IP Address: 61.255.71.181
HostName: 61.255.71.181
DShield Profile: Country:

Contact E-mail:

Total Records against IP:
13

Number of targets:
7

Date Range:
2002-06-07 to 2002-06-07

Ports Attacked (up to 10):

	Port
	Attacks

Fightback: not sent

Whois: IP Address : 61.255.71.0-61.255.71.255
Connect ISP Name : THRUNET
Connect Date : 20010808
Registration Date : 20011019
Network Name : THRUNET-CATV-SEOMYUNOWN

[Organization Information]
Orgnization ID : ORG205205
Name : THRUNET
State : SEOUL
Address : 1338-5 Seocho-2dong Seocho-ku, Seoul , Korea
Zip Code : 137-072

The fifth:

IP Address: 211.44.159.222
HostName: 211.44.159.222
DShield Profile: Country:
KR

Contact E-mail:
abuse_AT_hananet.net (bounced)

Total Records against IP:
1

Number of targets:
1

Date Range:
2002-06-22 to 2002-06-22

Ports Attacked (up to 10):

Port	Attacks
1433	1

Fightback: not sent

Whois: IP Address : 211.44.159.0-211.44.159.255
Connect ISP Name : HANANET
Connect Date : 20000209
Registration Date : 20000228
Network Name : BOOKBOOSO

[Organization Information]
Orgnization ID : ORG77097
Name : HANARO Telecom
State : SEOUL
Address : 1445-3 Seocho-Dong Seocho-Ku
Zip Code : 137-728

And the last IP address:

IP Address: 61.82.129.247
HostName: 61.82.129.247
DShield Profile: Country:

Contact E-mail:

Total Records against IP:
5239

Number of targets:
4698

Date Range:
2002-06-21 to 2002-06-21

Ports Attacked (up to 10):

Port	Attacks
1433	
545	

Fightback: not sent

Whois:
```
IP Address        : 61.82.114.0-61.82.135.255
Connect ISP Name  : KORNET
Connect Date      : 20011005
Registration Date : 20011005
Network Name      : KORNET-XDSL-YOUNGDONG

[ Organization Information ]
Orgnization ID    : ORG200062
Name              : YOUNGDONG NODE
State             : SEOUL
Address           : 675-4 YUKSAMDONG KANGNAMKU
Zip Code          : 135-080
```

Again, patterns emerge. First, all but one of the source addresses originate from Seoul, Korea. All but one have been reported as the origin of attacks on port 1433. The fourth IP address, 61.255.71.181, does not list specific port attack information but it does show 13 records against the IP address and 7 targets.

Now, let's get back to that time issue. The times reflected in the firewall logs are local to my network. A check of a world clock shows that Seoul is 16 hours ahead of me, which places most of these attacks in the afternoon and early evening. Not exactly prime time for most nefarious activities. It may be a stereotype but my firewall logs have shown me that a lot more scans originating from my time zone occur from 10:00pm through 2:00am.

Judging by the evidence at hand, I put my money on the Spida worm making the rounds in my subnet.

Correlations

These attack attempts were identified on my network by the host-based firewall, windump, and Snort running on 3 different hosts. Each capture is noted under "Event Traces," above.

In the wild, similar activity was reported on Incidents.org's Intrusions List on June 17, 2002, by Ken Connelly, Systems and Operations Manager, ITS - Network Services at the University of Northern Iowa in Cedar Falls, Iowa (relevant trace only)[8]:

```
The following extracts show the beginning and ending of scan activity
was [sic] detected on my network. The number following each set is
the total number of probes for that source. Timestamps are GMT-0500.

<snip>
```

[8] Connelly, Ken, "[LOGS] Summary of Large Scale Portscanning Detects," June 17, 2002. URL: http://cert.uni-stuttgart.de/archive/intrusions/2002/06/msg00155.html. September 7, 2002.

```
Jun 15 14:22:26 207.224.13.146:1318 -> xxx.yyy.1.2:1433 SYN ******S*
Jun 15 14:22:23 207.224.13.146:1319 -> xxx.yyy.1.3:1433 SYN ******S*
Jun 15 14:22:26 207.224.13.146:1320 -> xxx.yyy.1.4:1433 SYN ******S*
Jun 15 14:22:23 207.224.13.146:1321 -> xxx.yyy.1.5:1433 SYN ******S*
Jun 15 14:22:26 207.224.13.146:1322 -> xxx.yyy.1.6:1433 SYN ******S*
Jun 15 14:22:26 207.224.13.146:1323 -> xxx.yyy.1.7:1433 SYN ******S*
Jun 15 14:22:26 207.224.13.146:1324 -> xxx.yyy.1.8:1433 SYN ******S*
Jun 15 14:22:26 207.224.13.146:1325 -> xxx.yyy.1.9:1433 SYN ******S*
[...]
Jun 15 16:10:33 207.224.13.146:3947 -> xxx.yyy.255.251:1433 SYN
******S*
Jun 15 16:10:33 207.224.13.146:3948 -> xxx.yyy.255.252:1433 SYN
******S*
Jun 15 16:10:33 207.224.13.146:3940 -> xxx.yyy.255.244:1433 SYN
******S*
Jun 15 16:10:33 207.224.13.146:3944 -> xxx.yyy.255.248:1433 SYN
******S*
Jun 15 16:10:33 207.224.13.146:3949 -> xxx.yyy.255.253:1433 SYN
******S*
Jun 15 16:10:33 207.224.13.146:3941 -> xxx.yyy.255.245:1433 SYN
******S*
Jun 15 16:10:33 207.224.13.146:3937 -> xxx.yyy.255.241:1433 SYN
******S*
Jun 15 16:10:33 207.224.13.146:3945 -> xxx.yyy.255.249:1433 SYN
******S*
155097
<snip>
```

The traces captured by Ken Connolly represent a single infected host attempting to infect a subnet. This activity took place just four days prior to the scans captured on my network.

Similarly, Laurie Zirkle, from Communications Network Services at Virginia Tech, posted a great deal of port 1433 scans[9] to http://rdweb.cns.vt.edu/~lat/log_archives/ on the same days as the scans against my network (example excerpts only):

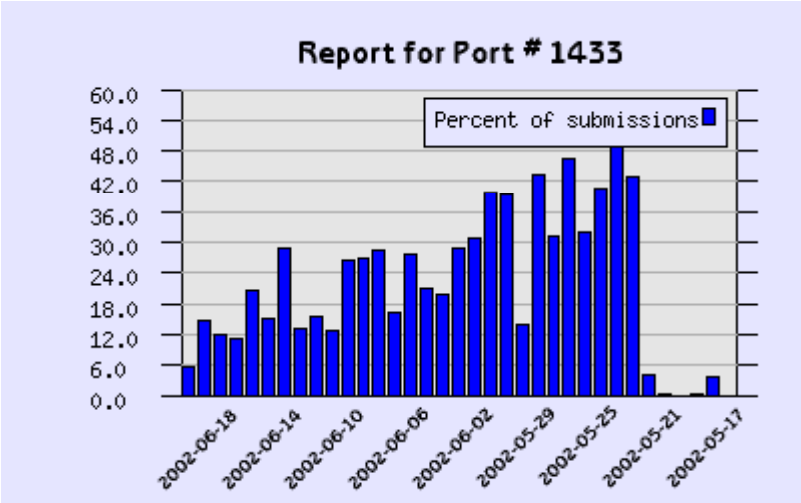
```
<snip>
Jun 19 05:13:18 211.216.46.211:2926 -> a.b.w.33:1433 SYN ******S*
Jun 19 05:13:18 211.216.46.211:2931 -> a.b.w.38:1433 SYN ******S*
Jun 19 05:13:18 211.216.46.211:2932 -> a.b.w.39:1433 SYN ******S*
Jun 19 05:13:18 211.216.46.211:2946 -> a.b.w.54:1433 SYN ******S*
Jun 19 05:13:19 211.216.46.211:2951 -> a.b.w.62:1433 SYN ******S*
<snip>

<snip>
Jun 20 06:59:01 203.231.235.145:3138 -> a.b.c.3:1433 SYN ******S*
Jun 20 06:59:01 203.231.235.145:3140 -> a.b.c.5:1433 SYN ******S*
Jun 20 06:59:01 203.231.235.145:3142 -> a.b.c.6:1433 SYN ******S*
Jun 20 06:59:02 203.231.235.145:3143 -> a.b.c.8:1433 SYN ******S*
Jun 20 06:59:02 203.231.235.145:3144 -> a.b.c.9:1433 SYN ******S*
<snip>

<snip>
Jun 21 04:56:55 196.44.140.69:3925 -> a.b.w.33:1433 SYN ******S*
Jun 21 04:56:55 196.44.140.69:3929 -> a.b.w.37:1433 SYN ******S*
Jun 21 04:56:55 196.44.140.69:3931 -> a.b.w.38:1433 SYN ******S*
```

[9] Zirkle, Laurie, "020619.txt," "020620.txt" and "020621.txt." June 2002 URL: http://rdweb.cns.vt.edu/~lat/log_archives/ (8 September 2002).

```
Jun 21 04:56:59 196.44.140.69:3945 -> a.b.w.53:1433 SYN ******S*
Jun 21 04:56:55 196.44.140.69:3946 -> a.b.w.54:1433 SYN ******S*
<snip>
```

These excerpts show that each attack host scanned entire subnets at a time.

The following port report generated by DShield.org shows the sharp rise and tapering of the worm activity:



Again, some of the activity may be attributed to a rekindled interest in the default SQL SA account. The gradual decrease is more than likely due to SQL system administrators changing their SA account passwords from the default, the easiest and most effective method to thwart the Spida worm.

As discussed previously, nearly all of the attacker IP addresses have been explicitly associated with attacks on port 1433. The two not previously discussed, 62.30.230.76 and 24.190.6.133, are shown below (relevant information only):

IP Address: 62.30.230.76
HostName: pc-62-30-230-76-sc.blueyonder.co.uk

DShield Profile: Country:
GB

Contact E-mail:
sb@cableinet.net

Total Records against IP:
31

Number of targets:
23

Date Range:
2002-06-11 to 2002-06-11

Ports Attacked (up to 10):

Port	Attacks

IP Address: 24.190.6.133
HostName: ool-18be0685.dyn.optonline.net

DShield Profile: Country:
US

Contact E-mail:
abuse@cv.net

Total Records against IP:
448

Number of targets:
393

Date Range:
2002-06-19 to 2002-06-19

Ports Attacked (up to 10):

Port	Attacks
1433	58

Once again, we see an IP address displaying unspecified complaints and

targets, and a known port 1433 attacker.

Evidence of Active Targeting

While the victim is, indeed, a Windows-based operating system, neither SQL nor Microsoft Database Engine (MSDE) are running. My network's external identity was more than likely targeted at random, the result of the worm's target subnet generation utility.

Severity

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

Criticality = 1
The scanned system is a Windows-based client workstation with very limited services available, none of which listen on port 1433.

Lethality = 5
Had this attack been successful, the attacker could potentially have gained administrative privileges to this system and to my network.

System Countermeasures = 5
The system scanned is fully patched, firewalled, and monitored by Snort and windump.

Network Countermeasures = 2
The perimeter firewall was disabled for this exercise. The router performs network address translation, which can obfuscate the internal network, but the router itself was vulnerable to attack, though not of this particular nature. Snort monitors the internal network, however, because it ran on a Windows machine for this exercise, the alerting feature was not available.

Therefore, (1 + 5) - (5 + 2) = Severity of -1

While the network defenses were weak, the host itself was ostensibly invulnerable because port 1433 was closed and SQL is not installed, therefore the SA account does not exist. The host is further protected by the firewall. Thus, the attack attempt failed.

Defensive Recommendations

The router performs network address translation and each internal host is firewalled and monitored. All systems, both Red Hat and Windows, are patched to current levels. The targeted service, SQL, is not running on any of my systems. Snort and windump run on two or more hosts at any given time, monitoring the internal network. Although the attack failed, it is still advisable to firewall the router to provide a perimeter defense.

Multiple Choice Question

The SQLSpida.B Worm is easily defeated by:

a) maintaining current patch levels
b) windump
c) Snort
d) changing the default SA account password

Answer: d
Microsoft SQL and MSDE database programs ship with a blank password for the SA account. SQLSpida.B infects machines by connecting to the SA account using the default password. Creating a password upon installation of SQL/MSDE effectively removes the vulnerability.

Detect #3 - Miscellaneous TCP Port 0 Traffic

This detect was downloaded from http://www.incidents.org/logs/Raw/ from the file, 2002.6.14. The logs have been sanitized to obfuscate the victim network(s). For this reason, I will be ignoring the "bad checksum," which appears in every packet in the logs. The architecture of the target network is unknown. I used SnortSnarf to identify the following traces.

I posted the following analysis to intrusions@incidents.org on September 18, 2002 under my personal email address of digigal11@hushmail.com. No comments, questions or other responses were received from the community. I reposted on October 14, 2002. I received two responses, which are addressed at the end of this detect analysis.

Event Traces

The following alerts were reported by Snort:

```
[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:01:20.744488 211.47.255.21:35917 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x1C2BF28D Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:01:23.734488 211.47.255.21:35917 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x1C2BF28D Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:01:29.734488 211.47.255.21:35917 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x1C2BF28D Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:01:41.734488 211.47.255.21:35917 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x1C2BF28D Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:01:52.744488 211.47.255.21:36282 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x1EA171C2 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:01:55.734488 211.47.255.21:36282 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x1EA171C2 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:02:01.734488 211.47.255.21:36282 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x1EA171C2 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:02:13.744488 211.47.255.21:36282 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x1EA171C2 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:02:24.744488 211.47.255.21:36687 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x208432CC Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:02:28.364488 211.47.255.21:36687 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x208432CC Ack: 0x0 Win: 0x16D0 TcpLen: 32
```

```
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:02:33.744488 211.47.255.21:36687 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x208432CC Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:02:45.744488 211.47.255.21:36687 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x208432CC Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:02:56.744488 211.47.255.21:37151 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x22603D38 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:02:59.744488 211.47.255.21:37151 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x22603D38 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:03:05.744488 211.47.255.21:37151 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x22603D38 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:03:17.744488 211.47.255.21:37151 -> 46.5.114.52:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x22603D38 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
```

Approximately four minutes later, the following alerts were reported:

```
[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:07:25.284488 211.47.255.20:45404 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x34154426 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:07:28.274488 211.47.255.20:45404 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x34154426 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:07:34.274488 211.47.255.20:45404 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x34154426 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:07:46.274488 211.47.255.20:45404 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x34154426 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
```

```
[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:07:57.284488 211.47.255.20:45768 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x35FF1AFC Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:08:00.274488 211.47.255.20:45768 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x35FF1AFC Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:08:06.274488 211.47.255.20:45768 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x35FF1AFC Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:08:18.274488 211.47.255.20:45768 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x35FF1AFC Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:08:29.274488 211.47.255.20:46121 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x3826D074 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:08:32.274488 211.47.255.20:46121 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x3826D074 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:08:38.274488 211.47.255.20:46121 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x3826D074 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:08:50.274488 211.47.255.20:46121 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x3826D074 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:09:01.274488 211.47.255.20:46508 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x39D826A3 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:09:04.274488 211.47.255.20:46508 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x39D826A3 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:09:10.274488 211.47.255.20:46508 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
```

```
******S* Seq: 0x39D826A3 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-18:09:22.274488 211.47.255.20:46508 -> 46.5.194.214:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x39D826A3 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
```

Approximately one hour and nine minutes later, the following alerts were reported:

```
[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:18:02.794488 211.47.255.23:52451 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x3F115C39 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:18:05.794488 211.47.255.23:52451 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x3F115C39 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:18:11.804488 211.47.255.23:52451 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x3F115C39 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:18:23.794488 211.47.255.23:52451 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x3F115C39 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:18:34.804488 211.47.255.23:52862 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x41106D9D Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:18:37.794488 211.47.255.23:52862 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x41106D9D Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:18:43.794488 211.47.255.23:52862 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x41106D9D Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:18:55.794488 211.47.255.23:52862 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x41106D9D Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:19:06.794488 211.47.255.23:53284 -> 46.5.193.73:0
```

```
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x42C7C674 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:19:09.794488 211.47.255.23:53284 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x42C7C674 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:19:15.794488 211.47.255.23:53284 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x42C7C674 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:19:27.794488 211.47.255.23:53284 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x42C7C674 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:19:38.794488 211.47.255.23:53664 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x441A7AD0 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:19:41.794488 211.47.255.23:53664 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x441A7AD0 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:19:47.794488 211.47.255.23:53664 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x441A7AD0 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-19:19:59.794488 211.47.255.23:53664 -> 46.5.193.73:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x441A7AD0 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
```

And finally, approximately two hours and fifty minutes later, the following alerts were generated:

```
[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:09:53.004488 211.47.255.22:39319 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x8C1D90B3 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:09:55.994488 211.47.255.22:39319 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x8C1D90B3 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
```

```
07/13-22:10:01.994488 211.47.255.22:39319 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x8C1D90B3 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:10:13.994488 211.47.255.22:39319 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x8C1D90B3 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:10:25.004488 211.47.255.22:39810 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x8D0EB8BE Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:10:27.994488 211.47.255.22:39810 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x8D0EB8BE Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:10:33.994488 211.47.255.22:39810 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x8D0EB8BE Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:10:45.994488 211.47.255.22:39810 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x8D0EB8BE Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:10:56.994488 211.47.255.22:40324 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x8F16D506 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:10:59.994488 211.47.255.22:40324 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x8F16D506 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:11:05.994488 211.47.255.22:40324 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x8F16D506 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:11:17.994488 211.47.255.22:40324 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x8F16D506 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:11:28.994488 211.47.255.22:40829 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x91D1CD2A Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
```

```
[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:11:31.994488 211.47.255.22:40829 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x91D1CD2A Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:11:37.994488 211.47.255.22:40829 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x91D1CD2A Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:1] MISC TCP port 0 traffic [**]
07/13-22:11:49.994488 211.47.255.22:40829 -> 46.5.148.130:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x91D1CD2A Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
```

Source of Trace

These traces were harvested from a Class B subnet, host and network specifics unknown.

Detect Generated By

These alerts were generated by Snort using the default rules. The capture binary was fed through SnortSnarf to generate the actual alerts.

Probability The Source Address Was Spoofed

The probability is low. As will be discussed below, this attack appears to be a reconnaissance effort using crafted packets. Such an effort indicates an interest in the responses. While the attacker may not legitimately own the scan sources, he or she had sufficient access to them to direct the attack.

Description of the Attack

This traffic appears to be a reconnaissance probe. In order to describe the attack, we need to look closely at the packets. Let's examine the first four packets sent from the source 211.47.255.21 to the target 46.5.114.52. The following is the result of running the binary capture through windump:

```
18:01:20.744488 211.47.255.21.35917 > 46.5.114.52.0: S
472642189:472642189(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale
0> (DF) (ttl 47, id 0, bad cksum e04c!)
0x0000       4500 0034 0000 4000 2f06 e04c d32f ff15 E..4..@./..L./..
0x0010       2e05 7234 8c4d 0000 1c2b f28d 0000 0000 ..r4.M...+......
0x0020       8002 16d0 51ca 0000 0204 05b4 0101 0402 ....Q...........
0x0030       0103 0300                               ....
18:01:23.734488 211.47.255.21.35917 > 46.5.114.52.0: S
472642189:472642189(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale
0> (DF) (ttl 47, id 0, bad cksum e04c!)
0x0000       4500 0034 0000 4000 2f06 e04c d32f ff15 E..4..@./..L./..
0x0010       2e05 7234 8c4d 0000 1c2b f28d 0000 0000 ..r4.M...+......
0x0020       8002 16d0 51ca 0000 0204 05b4 0101 0402 ....Q...........
0x0030       0103 0300                               ....
18:01:29.734488 211.47.255.21.35917 > 46.5.114.52.0: S
472642189:472642189(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale
0> (DF) (ttl 47, id 0, bad cksum e04c!)
0x0000       4500 0034 0000 4000 2f06 e04c d32f ff15 E..4..@./..L./..
0x0010       2e05 7234 8c4d 0000 1c2b f28d 0000 0000 ..r4.M...+......
0x0020       8002 16d0 51ca 0000 0204 05b4 0101 0402 ....Q...........
0x0030       0103 0300                               ....
18:01:41.734488 211.47.255.21.35917 > 46.5.114.52.0: S
472642189:472642189(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale
0> (DF) (ttl 47, id 0, bad cksum e04c!)
0x0000       4500 0034 0000 4000 2f06 e04c d32f ff15 E..4..@./..L./..
0x0010       2e05 7234 8c4d 0000 1c2b f28d 0000 0000 ..r4.M...+......
0x0020       8002 16d0 51ca 0000 0204 05b4 0101 0402 ....Q...........
0x0030       0103 0300                               ....
```

One point stands out right away, the destination port of "0." Port 0 is reserved, meaning it should not be used in any legitimate TCP or UDP connection. The hex value of "06" in the ninth byte clearly identifies the protocol as TCP. The 20-byte header size conforms to expected TCP behavior. With a destination of port 0, however, it may look like a duck, it may act like a duck, but it's clearly barking. Therefore, the origin of these packets is artificial.

The purpose of artificial, or "crafted," packets is typically to perform host or network reconnaissance. By behaving in an unexpected manner, the packet may slip past defenses configured to recognize only expected patterns or behaviors. Even if the packets are rejected, the host or network is giving the attacker information, namely, that there is a defense in place. The attacker can then alter the packets to test the sufficiency of those defenses or move on to the next target, in the hope that he or she will find a vulnerable host or network.

If the target responds, another reconnaissance task is accomplished by sending packets with unexpected or illogical features - identifying the operating system on the target host(s). Different operating systems react to unexpected requests in different ways. The attacker can match the response to the malformed request to an operating system's known response behaviors, then tailor their next step to the operating system.

In this case, it appears the targets did not respond. Note the repeating sequence number:

```
472642189:472642189(0)
```

Such repetition only occurs if the destination host does not respond. Had the target responded, the sequence number would increment.

I was unable to locate specific industry alerts regarding reconnaissance activities from recognized agencies such as the CERT Coordination Center. However, books such as *Hacking Exposed*[10] and *Network Intrusion Detection*[11] dedicate chapters to the ways and means to perform information gathering.

Attack Mechanism

These scans appear to be the work of hping or hping2 (hereinafter, collectively known as "hping"), a well-known freeware packet crafting utility, originally developed for the Linux operating system, later ported to Unix. Looking again at the windump traces above, we see the most obvious characteristics of hping right away, the previously discussed destination port of 0 in concert with the protocol setting of TCP. This is default behavior, as stated under "Protocol Selections" in the hping man page[12]:

> "Default protocol is TCP, by default hping2 will send tcp headers to target host's port 0...Often this is the best way to do an 'hide ping', useful when target is behind a firewall that drop [sic] ICMP."

Another point in favor of hping as the tool used in these attacks is the probability that the attack machines are running Linux. Taking a look at the headers on all of the packets, we see they all share the same TTL value, or Time To Live, of 47. I ran a traceroute from my own machine to the first attack source with the following result:

```
Tracing route to 211.47.255.21 over a maximum of 30 hops

  1    70 ms    30 ms    20 ms  xxx.yyy.zzz.1
  2    20 ms    20 ms    20 ms  aaa.bbb.ccc.ddd
  3    21 ms    20 ms    30 ms  eee.fff.ggg.hhh
  4    20 ms    20 ms    20 ms  iii.jjj.kkk.lll
  5    20 ms    20 ms    30 ms  mmm.nnn.ooo.ppp
  6    20 ms    20 ms    20 ms  qqq.rrr.sss.ttt
  7    20 ms    20 ms    30 ms  uuu.vvv.www.xxx
```

[10] McClure, Stuart; Scambray, Joel; Kurtz, George, Hacking Exposed (3d Ed.), Osborne/McGraw-Hill, 2001.

[11] Northcutt, Stephen; Novak, Judy; McLachlan, Donald, Network Intrusion Detection: An Analyst's Handbook (2d Ed.).New Riders Publishing, 2000.

[12] Sanfilippo, Salvatore, "HPING Man Page," date unknown.
URL: http://www.hping.org/manpage.html (10 September 2002)

```
 8    30 ms    20 ms    30 ms  zzz.aaa.bbb.ccc
 9    20 ms    30 ms    20 ms  ddd.eee.fff.ggg
10    20 ms    30 ms    20 ms  hhh.iii.jjj.kkk
11    20 ms    30 ms    30 ms  lll.mmm.nnn.ooo
12   181 ms   220 ms   190 ms  ppp.qqq.rrr.sss
13   180 ms   180 ms   180 ms  ttt.uuu.vvv.www
14   180 ms   180 ms   181 ms  x1x.y1y.z1z.a1a
15   180 ms   181 ms   180 ms  b1b.c1c.d1d.e1e
16     *        *        *     Request timed out.
17   210 ms   220 ms   221 ms  211.47.255.21

Trace complete.
```

Each hop decrements the originating TTL value by 1, ergo, adding 1 for each hop results in the original value of the TTL. Adding 17 to 47, the original TTL value becomes 64, which is the Linux default[13].

In this case, my machine was not the victim, nor am I able to determine the whereabouts of the victim hosts because their identities have been obfuscated so this particular exercise is purely speculative.

Turning our attention to another field in the packet headers lends additional credence to my theory. Let's examine the options field; again, the field is identical in each attack packet:

```
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
```

We see the option, Selective Acknowledgment, or SackOK. This is a signature of the Windows and Linux operating systems[14].

Why have I dismissed Windows as a possible attack operating system? Because the Windows default TTL value is 32. There is no logical way the TTL value of 47 could have come from a Windows box. Using hping, the attacker could have padded the value to add another layer of confusion to the packets, but the TTL field is as valuable to the attacker as to the analyst as a source of information.

Correlations

Scans or attacks conforming to the above are not well-documented, in terms of correlating reports or logs. However, in a firewall discussion group, I found the following, posted by Curt Wilson from the Southern Illinois University Credit Union (relevant statement in bold)[15]:

---

[13] Spitzner, Lance. "List of Fingerprints for Passive Fingerprint Monitoring," 23 May 2000. URL: http://project.honeynet.org/papers/finger/traces.txt (13 Sep. 2002)

[14] The Honeynet Project, Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of The Blackhat Community. Addison-Wesley, 2001. 99.

[15] Wilson, Curt, "Firewall Wizards: PIX Logs Source Port 0, TCP SYN/RST Scans?" 15 May 2001.

```
Detect 1: May 14 01:29:39 [192.149.115.1] %PIX-4-500004: Invalid
transport field for protocol=6, from 172.152.127.244/0 to
208.197.xxx.xx/1024

Invalid transport field. An AOL user attempts to connect from TCP
port 0 to TCP port 1024 on one of our IP addresses. Port 1024 has
been used for various types of trojan horse applications, but could
this also be an attack designed to look like response traffic to an
ephemeral port? Usually I see this type of traffic with a
destination of port 0, which seems to be used in operating
system fingerprinting techniques and nmap protocol scans. Perhaps
attacker is trying a different type of fingerprint by setting source
port instead?
<snip>
```

Not surprisingly, since no legitimate service is served, DShield's port reports turn up empty on searches for scans or attacks against port 0.

A look-up on the source IP addresses returned the following (contact information removed for brevity):

IP Address: 211.47.255.20
HostName: 211.47.255.20

DShield Profile: Country:
KR

Contact E-mail:
ip@saeroun.co.kr

Total Records against IP:
563

Number of targets:
49

Date Range:
2002-09-04 to 2002-09-04

Ports Attacked (up to 10):

Port	Attacks

Fightback: sent to ip@saeroun.co.kr on 2002-07-02 12:25:52
no reply received

URL: http://lists.insecure.org/firewall-wizards/2001/May/0052.html (10 September 2002).

Whois:
```
IP Address        : 211.47.255.0-211.47.255.255
Connect ISP Name  : SAEROUNNET
Connect Date      : 20000916
Registration Date : 20001002
Network Name      : ORG84651

[ Organization Information ]
Orgnization ID    : ORG100055
Name              : SAEROUNNET
State             : SEOUL
Address           : 789-28 sihungdong kumchungu
Zip Code          : 153-034
```

All of the attack machines belong to this same net block. The block owner, saeroun.co.kr, appears to be a Korean ISP with a web presence, located at http://www.saeroun.net.

While no specific port attacks are listed in the report, it's worth noting that there are 563 records against the IP range, and 49 targets.

Evidence of Active Targeting

The pattern of the scans implies that the victim hosts were not specifically targeted but, rather, they were unwitting participants in a subnet scan. Each attack host sent the same malformed packet to each intended victim 16 times. The timestamps indicate that all of the attacks occurred within a 4-hour window. The fact that there are multiple attackers gives the appearance of a coordinated attack, however, the fact that each packet sent from each attack host is identical seems indicative of a single source controlling the attack machines. The timing and number of attempts against each victim leads me to believe the single source unleashed an automated scan to machines he or she "owns," meaning machines previously comprised, in order to mask the attacker's true identity. As no legitimate service resides on port 0, the likelihood of a denial of service attempt is minimal.

The lack of a payload in the attack packets indicates that none of the packets attempted to actually do anything beyond eliciting response. As previously discussed, this is a known first step in gathering information about a range of hosts or a network, thus, it appears none of the victims were actively targeted.

Community Challenge

The following comments were posted by Robert Wagner on October 14, 2002:

"I saw mention of DoS attacks against UDP 0, TCP 0 on Google; can you find some and elaborate?"

The denial of service attacks directed toward port 0 are specific to CheckPoint FW-1. The exploit relates only to UDP packets on a VPN-1 that supports ISAKMP encryption.[16] The packets captured in this case were all TCP 0 which effectively eliminates that goal as a possibility.

The following comments were posted by Donald Smith on October 14, 2002:

"What kind of reconnaissance would send a nearly identical packet to the same host over and over again?"

As stated in the attack description, this form of recon is typically used for OS fingerprinting.

Mr. Smith further asked:

"What would be the hping command that would produce packets that match this?"

I am remiss; this is an obvious oversight on my part. Here is the basic command:
$ hping -k -w 5840 -M <sequence number> -S <destination IP address>

This command results in the following in my test lab:

```
10:17:55.756754 MY.NET.68.199.1108 > Tester3.0: S
472642189:472642189(0) win 5840
10:17:56.751720 MY.NET.68.199.1108 > Tester3.0: S
472642189:472642189(0) win 5840
10:17:57.751805 MY.NET.68.199.1108 > Tester3.0: S
472642189:472642189(0) win 5840
10:17:58.751878 MY.NET.68.199.1108 > Tester3.0: S
472642189:472642189(0) win 5840
```

The "-k" switch keeps the source port constant, the "-w 5840" sets the window size (more on this in a moment), the "-S" switch sets the SYN flag. In testing the command string, I realized that the window size had to have been set as the default is 512. This may have been done as an additional error check method to further pinpoint the target operating system, by seeing how it handled a large window size. Additionally, I realized that hping includes a switch to set the sequence number. Hence, the sequence number remaining static may also have been a manual addition rather than the firewall dropping the packet. Thus, unless I could correlate these events with the firewall logs, I have no proof that

[16] SecurityFocus Online, "Firewall-1 Port 0 Denial of Service Vulnerability," 9 August 1999. URL: http://online.securityfocus.com/bid/576/discussion/ (16 October 2002)

these packets were dropped or rejected.

The interval timing may have been accomplished with hping or, more likely, the attack was scripted. The latter would have been the easiest way to hit multiple targets in a short amount of time.

Finally, Mr. Smith commented on my statement as indicated by the initials (RS for me, DS for him):

RS> Whether there are firewalls on each host, or whether they all
RS> reside on a protected network is unknown. What is known is
RS> that the targets did not respond to malformed requests.

DS>I dont [sic] believe you have shown that.

As stated above, neither do I. I stand corrected. Whether or not these packets actually made it inside the target network is a question that is best answered by correlating the firewall logs with these events.

Severity

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

Criticality = 1
The nature of the victim machines is unknown. One aspect that is known is that any compromised machine can be used to attack others.

Lethality = 4
Whether the intent of the attacker was to elicit a response from the victim hosts or test their defenses, any response to the malformed packets would provide an attacker with information, which could be used to launch a targeted attack.

System Countermeasures = 5
Whether the packets were dropped as a result of host-based or perimeter firewalls is unknown. What is known is that the attacker gained no useful information.

Network Countermeasures = 5
Whether the packets were dropped as a result of host-based or perimeter firewalls is unknown. What is known is that the attacker gained no useful information.

Therefore, (1 + 4) - (5 + 5) = Severity of -5

Again, this equation may be incorrect as the nature of the victim host machines is unknown. The fact remains, though, that the victim hosts did not respond to the attack packets, thus the reconnaissance efforts failed.

Defensive Recommendations

Whether there are firewalls on each host, or whether they all reside on a protected network is unknown. What is known is that the targets did not respond to malformed requests. Without more information on the victim hosts, I am able to report only on limited facts at hand. As a matter of practice, I recommend both perimeter and host-based monitoring and protection, disabling unnecessary services, and keeping up to date on patches and security fixes, regardless of the role or operating system of any machine.

Multiple Choice Question

A "crafted" packet can be used to:

a. Identify an operating system
b. Defeat intrusion monitoring systems
c. Inject malicious data
d. All of the above

Answer: d
A constructed packet can mimic legit traffic as a transport method for malicious code, or contain features of legitimate traffic with unexpected characteristics to either bypass protections based on expected packet characteristics or to elicit a response which can be used to identify the target's operating system.

References

Northcutt, Stephen; Cooper, Mark; Fearnow, Matt; Frederick, Karen, Intrusion Signatures and Analysis. New Riders Publishing, 2001.

Evans, Chris, "Format Strings: bug #2 LPRng." Bugtraq post. 26 Sep 2000. URL: http://online.securityfocus.com/archive/1/85002 (8 Sep. 2002).

Connelly, Ken, "[LOGS] Summary of Large Scale Portscanning Detects," 17 June 2002
URL: http://cert.uni-stuttgart.de/archive/intrusions/2002/06/msg00155.html (7 September 2002)

Zirkle, Laurie, scan logs, June 2002
URL: http://rdweb.cns.vt.edu/~lat/log_archives/020623.txt (7 September 2002)

Bakos, George, "SQLsnake Code Analysis," 21 May 2002.
URL: http://www.incidents.org/diary/diary.php?id=157 (7 September 2002)

Zirkle, Laurie, "020619.txt," "020620.txt" and "020621.txt." June 2002
URL: http://rdweb.cns.vt.edu/~lat/log_archives/ (8 September 2002).

McClure, Stuart; Scambray, Joel; Kurtz, George, Hacking Exposed (3d Ed.). Osborne/McGraw-Hill, 2001.

Northcutt, Stephen; Novak, Judy; McLachlan, Donald, Network Intrusion Detection: An Analyst's Handbook (2d Ed.).New Riders Publishing, 2000.

Sanfilippo, Salvatore, "HPING Man Page," date unknown.
URL: http://www.hping.org/manpage.html (10 September 2002)

Spitzner, Lance. "List of Fingerprints for Passive Fingerprint Monitoring," 23 May 2000.
URL: http://project.honeynet.org/papers/finger/traces.txt (13 Sep. 2002)

The Honeynet Project, Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of The Blackhat Community. Addison-Wesley, 2001. 99.

Wilson, Curt, "Firewall Wizards: PIX Logs Source Port 0, TCP SYN/RST Scans?" 15 May 2001.
URL: http://lists.insecure.org/firewall-wizards/2001/May/0052.html (10 September 2002).

SecurityFocus Online, "Firewall-1 Port 0 Denial of Service Vulnerability," 9 August 1999.
URL: http://online.securityfocus.com/bid/576/discussion/ (16 October 2002

Assigment #3 - Analyze This

Executive Summary

Participation in "peer to peer" file sharing networks, (hereinafter "P2P"), is a recognized risk to any internal network. On this campus network, P2P presents a tenable threat to the integrity, confidentiality, and availability of the university's resources. An analysis of five days' worth of logs harvested from the campus network revealed a saturation of P2P traffic to and from external sources.

A thorough forensic analysis should be performed on the following servers to determine the nature and volume of files being shared on multiple P2P networks:

MY.NET.70.200
MY.NET.70.207
MY.NET.81.27
MY.NET. 82.2
MY.NET. 83.150
MY.NET.88.162
MY.NET.111.198
MY.NET.137.18
MY.NET.163.107
MY.NET.165.24

In addition, the following Windows' Internet Information System web servers should rebuilt and brought up to current security patch level due to NIMDA worm infection:

MY.NET.53.45
MY.NET.84.234
MY.NET.100.208
MY.NET.153.171
MY.NET.168.13

Host Overview

By reviewing the scan, alert and out of spec logs, I gleaned the following:

MY.NET.6.7 - POP3
MY.NET.6.40 - mail server
MY.NET.70.34 - AFS fileserver
MY.NET.70.200 - Network UPS Tools
MY.NET.70.207 - Proxy server

MY.NET.82.2 - Proxy server
MY.NET.84.233 - Windows file server
MY.NET.84.244 - Samba file server
MY.NET.84.246 - Windows file server
MY.NET.84.247 - Samba server
MY.NET.84.254 - Samba server
MY.NET.87.50 - 3Com remote access server (AccessBuilder)
MY.NET.100.158 - DNS server
MY.NET.100.165 - web server
MY.NET.100.208 - web server
MY.NET.100.217 - mail server
MY.NET.136.3 - Samba server
MY.NET.136.8 - Windows file server
MY.NET.136.9 - Windows file server
MY.NET.136.18 - Windows file server
MY.NET.137.7 - DNS
MY.NET.140.179 - Windows file server
MY.NET.183.33 - RPC
MY.NET.179.77 - web server

This list is not exhaustive. The network appears to be a Class B, comprised of several hundred servers across multiple 0/24 subnets. The above list was compiled by identifying the services from the scan logs, cross-referencing alerts, and observing details such as time of day. The latter is relevant in cases such as MY.NET.70.207, which did double duty, a proxy server by day, serving client requests on CDL ports, and a game server by night, receiving hundreds of connections from Medal of Honor ports 12300/UDP and 12203/UDP.

Additional observations:

- MY.NET.82.2 appears to be a second proxy by day and game server by night, similar to MY.NET.70.207.

- MY.NET.70.34 participated in distributed file sharing via AFS on UDP port 7001 with 8 other educational institutions. These account for a large number of logged scans.

- MY.NET.70.200 connections (3493/UDP) were made using the Network UPS Tools' browser interface. The clients all connected from port 80.

- It is safe to say that the university does not use Microsoft's SQL server. There was a high volume of 1433/TCP SYN scans which all came from outside networks. Unlike the NIMDA scans which came in from external sources on 80/TCP and were later followed by SYN scans from the internal hosts listed above to external networks on 80/TCP, indicating infection. This is detailed below.

Files Analyzed

The following logs were downloaded, providing the basis for this analysis:

alert.020801	scans.020801	oos_Aug.1.2002
alert.020802	scans.020802	oos_Aug.2.2002
alert.020803	scans.020803	oos_Aug.3.2002
alert.020804	scans.020804	oos_Aug.4.2002
alert.020805	scans.020805	oos_Aug.5.2002

Detects Prioritized By Severity

The alert logs were fed into SnortSnarf, the results of which were compiled into a spreadsheet by day and alert type. This provided me with a timeline with which I could clearly assess the impact and risks to the network's resources, correlating activities with the scan logs. Upon analysis of the results, I was able to break out the alerting activities into categories of high, medium, and low impact. These categories are defined as:

- High: immediate threat to resource availability, integrity or confidentiality; successful system compromise; existence of targeted services; low false positive potential
- Medium: potential but unsubstantiated threat to resource availability, integrity or confidentiality; information gathering; targeted services do not exist
- Low: no potential threat to resource availability, integrity or confidentiality; targeted services do not exist; alerts generated by legitimate traffic

Each of the detects below fell into "high" category.

Detect #1 - NIMDA Alert

There were over 1,000,000 NIMDA alerts in a single day. Overall, NIMDA scans accounted for the majority of scans logged over the 5-day period.

NIMDA is a particularly virulent self-propagating worm that spreads in one of five methods:

- Via email as an attachment named, "readme.exe"
- Appending to HTM, HTML and ASP files on infected hosts
- Exploiting the IIS Extended Unicode Directory Traversal or escaped characters decoding vulnerabilities

- Via open network shares, creating or replacing files such as admin.dll, riched20.dll, or readme.eml
- By accessing the "root.exe" or "cmd.exe" backdoor left behind by Code Red II infection

The propagation methods exploit these known Microsoft vulnerabilities:

- IIS/PWS Escaped Characters Decoding Command Execution
- IE MIME Header Attachment Execution
- IIS/PWS Extended Unicode Directory Traversal
- Microsoft Office 2000 DLL Execution

As far as inflicting damage, NIMDA's bag of tricks includes (but is not limited to):

- the creation or activation of disabled Guest accounts then granting it administrative privileges, thus giving the worm's creator root privileges
- granting full control to the Everyone group on a C: share, which gives anyone, including remote users, access to system files
- replacing executables it finds on open network shares with infected files of the same name
- consuming network bandwidth and resource utilization in its attempts to further propagate

Scans from external victims occurred every day, apparent by a single source IP address sending SYN packets to port 80/TCP on every host on a MY.NET subnet, e.g:

<snip>
```
Aug  2 06:07:26 217.228.34.247:3711 -> MY.NET.5.83:80 SYN ******S*
Aug  2 06:07:26 217.228.34.247:3715 -> MY.NET.5.87:80 SYN ******S*
Aug  2 06:07:26 217.228.34.247:3718 -> MY.NET.5.90:80 SYN ******S*
Aug  2 06:07:26 217.228.34.247:3727 -> MY.NET.5.99:80 SYN ******S*
Aug  2 06:07:26 217.228.34.247:3729 -> MY.NET.5.101:80 SYN ******S*
Aug  2 06:07:26 217.228.34.247:3731 -> MY.NET.5.103:80 SYN ******S*
Aug  2 06:07:26 217.228.34.247:3732 -> MY.NET.5.104:80 SYN ******S*
Aug  2 06:07:26 217.228.34.247:3733 -> MY.NET.5.105:80 SYN ******S*
Aug  2 06:07:26 217.228.34.247:3734 -> MY.NET.5.106:80 SYN ******S*
Aug  2 06:07:26 217.228.34.247:3735 -> MY.NET.5.107:80 SYN ******S*
```
<snip>

The following internal hosts exhibited similar behavior within hours of their respective subnets being scanned by external victims:

MY.NET.53.45
MY.NET.84.234
MY.NET.100.208
MY.NET.153.171

MY.NET.168.13

In other words, each of these hosts sent SYN packets to port 80/TCP on every live host on multiple subnets, both internal and external, in rapid succession. The impact of scanning activity alone likely resulted in a network bandwidth denial of service or significant reduction of availability of resources provided by these victims. The alerts themselves may well have also resulted in a network slowdown due to the incredibly high volume.

These infections may be the result of the network's exposure to P2P networks. Such networks are known breeding grounds for viruses and worms, due to the ease of propagation vectors. In this case, the most significant exposure came by way of network shares open to external networks. The "Top Talkers" section below contains a detailed discussion of this exposure.

Out of curiosity, I looked up the first source of NIMDA scanning. The "whois" output revealed the source as a dial-up account in Germany:

```
inetnum:      217.224.0.0 - 217.237.161.47
netname:      DTAG-DIAL15
descr:        Deutsche Telekom AG
country:      DE
admin-c:      DTIP-RIPE
tech-c:       ST5359-RIPE
status:       ASSIGNED PA
remarks ***************************************************************
remarks * ABUSE CONTACT: abuse@t-ipnet.de IN CASE OF HACK ATTACKS, *
remarks * ILLEGAL ACTIVITY, VIOLATION, SCANS, PROBES, SPAM, ETC.   *
remarks ***************************************************************
notify:       auftrag@nic.telekom.de
notify:       dbd@nic.dtag.de
mnt-by:       DTAG-NIC
changed:      auftrag@nic.telekom.de 20020108
source:       RIPE

route:        217.224.0.0/11
descr:        Deutsche Telekom AG, Internet service provider
origin:       AS3320
mnt-by:       DTAG-RR
changed:      bp@nic.dtag.de 20010405
source:       RIPE

person:       DTAG Global IP-Adressing
address:      Deutsche Telekom AG
address:      Bayreuther Strasse 1
address:      D-90409 Nuernberg
address:      Germany
phone:        +49 911 68909856
e-mail:       ripe.dtip@telekom.de
nic-hdl:      DTIP-RIPE
mnt-by:       DTAG-NIC
changed:      ripe.dtip@telekom.de 20020717
source:       RIPE

person:       Security Team
address:      Deutsche Telekom AG
address:      Technikniederlassung Schwaebisch Hall
```

```
address:        D-89070 Ulm
address:        Germany
phone:          +49 731 100 84055
fax-no:         +49 731 100 84150
e-mail:         abuse@t-ipnet.de
nic-hdl:        ST5359-RIPE
notify:         auftrag@nic.telekom.de
notify:         dbd@nic.dtag.de
mnt-by:         DTAG-NIC
changed:        auftrag@nic.telekom.de 20010321
source:         RIPE
```

Recall the methods that NIMDA attacks. Random IP generation is not one of them. Therefore, this machine had to "know" about the campus subnets somehow. The fact that the attack originated from Germany casts doubt on the possibility that the attacker was or is a student. More likely, the machine belongs to a P2P user and the worm used the directory traversal method to exploit its open shares, thus scanning for additional victims, such as the campus hosts.

It's worth noting that Microsoft released patches for all of the vulnerabilities exploited by NIMDA by August 2001[17][18]. The original CERT Advisory on NIMDA, CA-2001-26, was released September 18, 2001. The scans and alerts seen in the campus logs follow the "textbook" definition of NIMDA probes[19]:

> "To briefly summarize what Nimda does...(i)t probes each IP address within a randomly-selected range of IP addresses, attempting to exploit weaknesses that, unless already patched, are known to exist in computers with Microsoft's Internet Information Server."

A review of the archives of the Security Focus Incidents mailing list revealed steady and constant NIMDA scans around the country for the last year. Posters reported seeing behavior similar to the scans and alerts contained in the campus logs. Unfortunately, none of the posters included their logs, merely providing narrative correlation, such as this, from Brian Mooney, posted February 26, 2002[20]:

> "I have been seeing those scans pretty nonstop since the outbreak of Nimda. AT&T tells me that they have blocked Code Red, CRII, and

---

[17] Microsoft TechNet, "Microsoft Security Bulletin MS01-020," 29 March 2001
URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-020.asp (10 October 2002)

[18] Microsoft TechNet, "Microsoft Security Bulletin MS01-044," 15 August 2001
URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-044.asp (10 October 2002)

[19] Author unknown, "Nimda," 20 September 2001.
URL: http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci770982,00.html (14 October 2002)

[20] Mooney, Brian, "Re: Wave of Nimda-like Hits This Morning?" 26 February 2002
URL: http://www.der-keiler.de/Mailing-Lists/securityfocus/incidents/2002-02/0212.html (14 October 2002)

Nimda upstream, but I still get this traffic 15 times a day or so. Yesterday,
I had one IP hit my machine, looking for cmd.exe 27 times..."

I recommend that each of the servers exhibiting NIMDA infection-like behavior be taken offline immediately. A forensic analysis would provide insight to the exact vulnerability exploited. A complete rebuild of each system is also recommended, bringing them up to current patch and hot fix levels before reconnecting to the campus network.

Detect #2 - IIS Unicode Attack

There were 493,490 Unicode alerts generated. These alerts are likely a by-product of the NIMDA activity. As noted above, one of the methods of propagation is the IIS Unicode Directory Traversal exploit.

The attack consists of the inclusion of a Unicode character, "/" or "\," with extended code which may enable an anonymous user to read or write to files or execute commands. By including the ".." file system navigational technique, vulnerable Microsoft's IIS web servers read ".." in a code string placed in a URL as a request to back out of the current directory, allowing attackers into the web root, or other specifically requested directories.

The vulnerability is discussed in detail in CERT's Vulnerability Note VU#111677.

The alert logs show 798 sources for the Unicode attack, 303 of which were internal hosts. Each of the internal hosts listed as NIMDA scan sources above was also a source of Unicode alerts.

Looking at the Snortsnarf output, the rules differentiating NIMDA alerts from IIS Unicode attacks appear to be based on payload. Each of the rules appears to be custom; a review of the default rules included with Snort version 1.9 and the signature database found on the Snort website at http://www.snort.org did not reveal output messages exactly like the alerts. Therefore, I deduced the rules were edited or, in the case of NIMDA, custom written. NIMDA alerts are worded as:

```
08/05-21:30:54.598812 [**] NIMDA - Attempt to execute root from
campus host **] MY.NET.100.208:2886 -> 46.167.57.32:80
```

A second alert type substituted the "cmd" for "root." This appears to be identifying NIMDA on the basis of the attempt to access the Code Red II backdoors, discussed more fully in Detect #3.

Here's an example of the IIS Unicode alert:

```
08/05-21:21:55.988472 [**] spp_http_decode: IIS Unicode attack
```

detected [**] MY.NET.100.208:2150 -> 130.199.172.67:80

The "spp_http_decode" in the message is similar to default rule configuration, which leads me to believe it was simply an edited signature. Given the alert language, I imagine the rules looked something like this:

NIMDA
alert tcp any -> any 80 (msg:"NIMDA - Attempt to execute cmd.exe from campus host"; flags: A+; uricontent::"cmd.exe";nocase; classtype: attempted-user;)

A second rule must have been written to substitute "root.exe" for "cmd.exe."

IIS Unicode
alert tcp any -> any 80 (msg: "spp_http_decode: IIS Unicode attack detected"; flags: A+; uricontent: "..|25|c1|25|1c"; nocase; classtype: system-attempt;)

Thus, the same host could trigger both the Nimda and IIS Unicode alerts if the scan payloads contained both the command or root execution attempt and the directory traversal attempt.

This particular vulnerability was first patched by Microsoft in August 2000,[21] subsequently included in a security roll-up patch released in August 2001, as well as Service Pack 2. I strongly recommend that all servers in the campus network be brought to each vendor's current patch level.

Detect #3 - ISAPI Overflow

This attack accounted for 482,333 alerts. Most of the alerts originated from a single internal source, MY.NET.84.234. This same machine also generated a great deal of NIMDA alerts, as noted above.

The ISAPI overflow refers to a Microsoft IIS buffer overflow, which allows embedded code to run with system-level privileges. While many exploits attempt to use this as an attack method, the most successful is Code Red; the vulnerability is the heart and soul of Code Red and later variants. Its method of propagation is a malformed GET request to an IIS web server, which includes a buffer overflow to an Indexing Server Application Program Interface ("ISAPI") extension, .idq.dll, that gets installed with IIS. This .dll supports the Internet data administrative script files, aka ".ida" and the Internet data queries, aka ".idq."

It's on these file extensions that the ISAPI overflow signatures are based. The alerts quote the arachNIDS database of Snort detection signatures. For example:

[21] Microsoft TechNet, "Microsoft Security Bulletin MS00-057," 10 August 2000. URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms00-057.asp (14 October 2002)

```
08/04-17:26:26.356846 [**] IDS552/web-iis_IIS ISAPI Overflow ida
INTERNAL nosize [**] MY.NET.84.234:1069 -> 96.161.223.46:80
```

"IDS552" is the arachNIDS signature number for the buffer overflow specific to the .ida extension, which is the alert seen in the campus logs:

alert TCP $EXTERNAL any -> $INTERNAL 80 (msg: "IDS552/web-iis_IIS ISAPI Overflow ida"; dsize: >239; flags: A+; uricontent: ".ida?"; classtype: system-or-info-attempt; reference: arachnids,552;)

Code Red was initially detected in the summer of 2001. It hasn't stopped circulating since. The Internet Storm Center reports on monthly spikes in scanning activity on port 80[22] at the same time every month due to the worm's built-in timer; it "wakes up" around the 20th. A variant infection, Code Red II, moves the cmd.exe file to the publicly accessible IIS Scripts and MSADC directories. This allows the worm's author, or anyone else, to execute commands with the IIS server-level permissions. As mentioned previously, NIMDA looks for this backdoor.

This machine may well be the catalyst for the campus NIMDA outbreak. The amount of ISAPI Overflow alerts generated almost certainly means the machine was first compromised by Code Red or Code Red II before NIMDA took hold.

Once again, Microsoft has had a patch available to mitigate this vulnerability since June 2001[23]. I strongly recommend that MY.NET.84.243 be removed from the network, rebuilt and brought up to the current security patch level before being put back on the live network.

Detect #4 - UDP Source and Destination Outside Network

This alert was triggered 106,853 times, all originating from 4 source subnets. Based on the number of alerts, I initially placed this detect in the "high" category. Upon closer examination, I deduced that all of the alerts are false positives, attributable to several separate legitimately occurring events.

First, there were several daily occurrences of this:

```
08/04-00:00:01.480815 [**] UDP SRC and DST outside network [**]
3.0.0.99:137 -> 10.0.0.1:137
```

The source address turned out to be General Electric:

---

[22] Internet Storm Center "Current Status" banner, updated daily.
URL: http://isc.incidents.org (20 October 2002)

[23] Microsoft TechNet, "Microsoft Security Bulletin MS01-033," 18 June 2001.
URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp (10 October 2002)

```
Search results for: 3.0.0.9


OrgName:    General Electric Company
OrgID:      GENERA-9

NetRange:   3.0.0.0 - 3.255.255.255
CIDR:       3.0.0.0/8
NetName:    GE-INTERNET
NetHandle:  NET-3-0-0-0-1
Parent:
NetType:    Direct Assignment
NameServer: ns.ge.com
NameServer: ns1.ge.com
NameServer: ns2.ge.com
Comment:
RegDate:    1988-02-23
Updated:    2002-09-26

TechHandle: GET2-ORG-ARIN
TechName:   General Electric Company
TechPhone:  +1-518-612-6672
TechEmail:  genictech@ge.com
```

The fact that the destination is 10.0.0.1, an internal IP address, leads me to believe that this is a response to an internally generated request. The fact that the request is made to an electricity provider on a daily basis, leads me to believe that it could be a health check on university's power systems.

Next, there were daily occurrences of alerts originating from 128.223.75.xx, where the last octet varied:

```
08/03-19:24:23.553087 [**] UDP SRC and DST outside network [**]
128.223.75.157:1346 -> 229.55.150.208:1345
```

A lookup of the source address revealed it to be a university:

```
Search results for: 128.223.75.157


OrgName:    University of Oregon
OrgID:      UNIVER-193

NetRange:   128.223.0.0 - 128.223.255.255
CIDR:       128.223.0.0/16
NetName:    UONET
NetHandle:  NET-128-223-0-0-1
Parent:     NET-128-0-0-0-0
NetType:    Direct Assignment
NameServer: PHLOEM.UOREGON.EDU
NameServer: ARIZONA.EDU
NameServer: RUMINANT.UOREGON.EDU
NameServer: DNS.CS.UOREGON.EDU
Comment:
RegDate:
Updated:    1996-08-27

TechHandle: DMM65-ARIN
TechName:   Meyer, David
```

```
TechPhone:  +1-541-915-0094
TechEmail:  dmm@antc.uoregon.edu
```

The destination address is reserved by IANA for multicasts, as specified in RFC 3171. These are likely streaming media broadcasts, such as video conferencing or distance learning, either of which is common to a university environment. The number of alerts suggests the number of connections, i.e., participants.

The third occurrence is:

```
08/01-07:15:03.298135 [**] UDP SRC and DST outside network [**]
63.250.213.12:1031 -> 233.28.65.148:5779
```

Again, the last octet in each varies. And once again, I believe this traffic is another example of streaming media broadcasts. The source address is Yahoo!'s broadcast service:

```
Search results for: 63.250.213.12


OrgName:    Yahoo! Broadcast Services, Inc.
OrgID:      YAHO

NetRange:   63.250.192.0 - 63.250.223.255
CIDR:       63.250.192.0/19
NetName:    NETBLK2-YAHOOBS
NetHandle:  NET-63-250-192-0-1
Parent:     NET-63-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.YAHOO.COM
NameServer: NS2.YAHOO.COM
NameServer: NS3.YAHOO.COM
NameServer: NS4.YAHOO.COM
NameServer: NS5.YAHOO.COM
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    1999-11-24
Updated:    2002-03-27

TechHandle: NA258-ARIN
TechName:   Netblock Admin, Netblock
TechPhone:  +1-408-349-7183
TechEmail:  netblockadmin@yahoo-inc.com
```

The destination is another IANA block reserved for multicasts using GLOP addressing. Correlation for this particular multicast was found in my fellow analyst, Bree Elliott's (#0400) analysis[24].

Finally, alerts originating from 169.254.17.xx occurred daily:

```
08/01-09:43:06.237039 [**] UDP SRC and DST outside network [**]
169.254.17.23:3350 -> 239.255.255.250:1900
```

[24] Elliott, Bree. "GIAC Certification Practical, SANS Online Curriculum, v. 2.9." 2001. URL: http://www.giac.org./practical/Bree_Elliott_GCIA.doc. Analyze This, page 35. (16 October 2002)

The source:

```
Search results for: 169.254.17.23


OrgName:    Internet Assigned Numbers Authority
OrgID:      IANA

NetRange:   169.254.0.0 - 169.254.255.255
CIDR:       169.254.0.0/16
NetName:    LINKLOCAL
NetHandle:  NET-169-254-0-0-1
Parent:     NET-169-0-0-0-0
NetType:    IANA Special Use
NameServer: BLACKHOLE-1.IANA.ORG
NameServer: BLACKHOLE-2.IANA.ORG
Comment:    Please see RFC 3330 for additional information.
RegDate:    1998-01-27
Updated:    2002-10-14

OrgTechHandle: IANA-ARIN
OrgTechName:   Internet Corporation for Assigned Names and Number
OrgTechPhone:  +1-310-823-9358
OrgTechEmail:  res-ip@iana.org
```

The destination is also another IANA block reserved for multicasts. The source IP address range belongs to what is known as a "link local block." The range is allocated between hosts on a single network segment, distributed to hosts by auto-configuration in the absence of a DHCP server. Again, the destination address range as a multicast block suggests normal traffic for a university segment, whereby students or conference participants received an IP address upon joining.

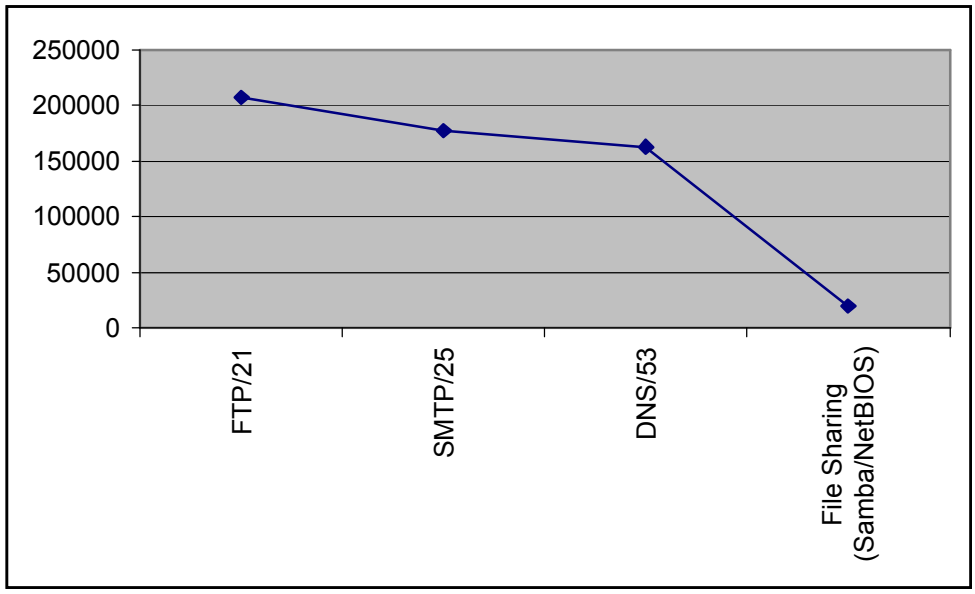
Top Talkers

As noted throughout this analysis, P2P and Internet game services are prevalent on the campus network. The following internal hosts participate as indicated:

Host Name	P2P Program	Number of Connections
MY.NET.70.200	Blubster	2,436,160
MY.NET.70.207	Medal of Honor (game)	137,261
MY.NET.81.27	MSN Game Zone	31,930
MY.NET.82.2	Medal of Honor (game)	127,810
MY.NET.83.150	WinMX	89,932
MY.NET.88.162	KaZaa	43
MY.NET.111.198	eDonkey2000	57
MY.NET.137.18	Gnutella	2,398
MY.NET.163.107	Gnutella	1,076
MY.NET.165.24	WinMX	104,415

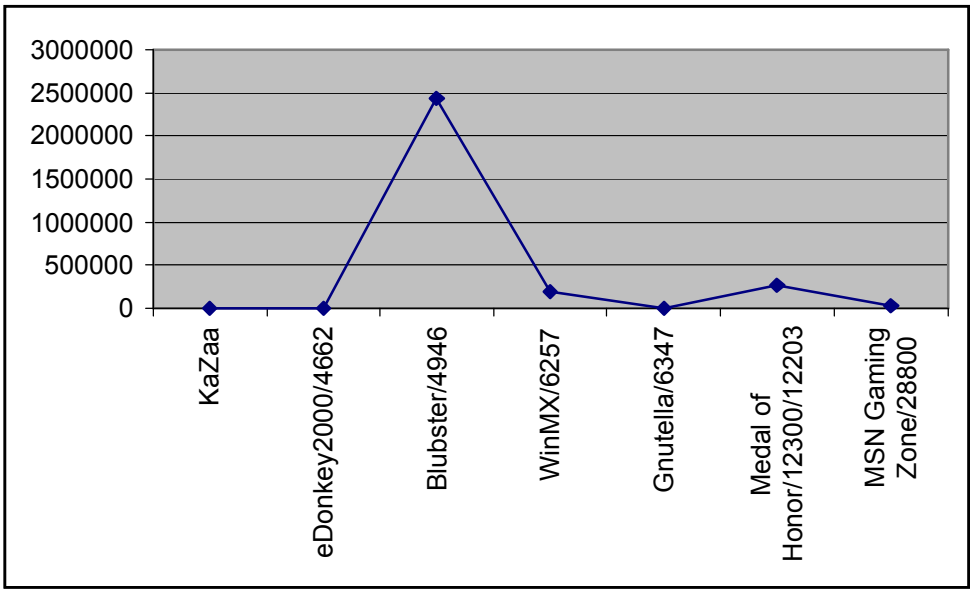
I've included the hosts who appear to be minor players for two reasons. First, even one unprotected open network share results in a major exposure to the internal network as illustrated by the NIMDA infections. Second, the P2P programs they host are major players in the wild. Universities around the country have begun limiting such programs, naming KaZaa, specifically.[25] Users who sniff their file trades in progress will notice the ".edu" on these hosts. Educational institutions are widely known for their lack of security. When - not if - external users on KaZaa, Gnutella and eDonkey become aware of the existence of these servers, the amount of traffic will increase dramatically. Since KaZaa is currently the most well-known of the file-sharing P2P networks, with literally millions of users, I can only assume MY.NET.88.162 was a recent addition to the network. Indeed, the connections on this machine, as well as MY.NET.111.198 and MY.NET.163.107, appeared in the final hours on the last day of the scan logs reviewed.

To illustrate the impact of P2P on the campus network, consider the graphs below. The first graph shows the total connections to legitimate services for the 5-day period:



Second, we see the total connections to the P2P and game services:

[25] Arana, Gail, "Yale Limits Use of Kazaa, Other Clients," 15 October 2002. URL: http://www.yaledailynews.com/article.asp?AID=20150 (16 October 2002).



Note the additional decimal "0" in the lower graph. While most of the P2P traffic is equal to or slightly less than the legitimate services, Medal of Honor, WinMX, and certainly Blubster account for as much, or in the case of Blubster, ten times the amount connections. The direct affect to the network is the demand on bandwidth utilization.

Other chief concerns are the amount of disk space unwittingly dedicated to audio, video and software files stored on participating servers by P2P users. Given the volume of connections, it's safe to assume the storage utilization is into the gigabytes. The possibility of virus- or worm-infected files, trojan horses and malware being stored on these servers is a concern, as well.

Finally, there exists the potential that the university may be considered an accomplice in piracy. File trading on P2P networks exposes the university to anti-piracy lawsuits. The precedent was set in the lawsuit filed by performers Metallica and Dr. Dre against Napster in 2000, in which several universities were named defendants.[26]

Recommendations

As noted throughout this analysis, I strongly recommend that virus-infected hosts be removed from the network immediately. Each should be rebuilt and brought to current security patch or hotfix level before being put back on the network.

---

[26] Borland, John, "Hollywood Chases Down Campus Pirates," 10 October 2002. URL: http://news.com.com/2100-1023-961637.html (16 October 2002)

Further, each of the hosts listed in the Top Talkers section should be removed from the network and forensically examined to determine the nature and extent of files being shared. Once all unauthorized data has been removed and/or remediated, the hosts may be placed back on the network.

Mitigating the participation of campus hosts on P2P networks is as easy as blocking the relevant ports at the firewall, both ingress and egress, unless specifically required. Snort rules should be added to watch for suspect connection attempts originating from inside the campus network to determine if any additional hosts are involved.

Finally, the university should adopt a proper use policy, clearly defining authorized activities and strictly prohibiting students from using campus resources for unauthorized data storage and file sharing.

Description of Analysis Process

The files used to compile this analysis were downloaded from http://www.incidents.org/logs. Following the example of my fellow analysts, I attempted to concatenate the alert logs into a single file for processing with SnortSnarf. However, the volume of pre-sorted alerts was over 2 gigabytes and the process utilization crashed every server I tried, both Solaris and Windows 2000, each with a 1Ghz processor and anywhere from 512MB to 1GB of RAM. I finally settled for SnortSnarfing the first 3 days' alert logs individually on the Windows server. The last two days were, by far, the largest, each containing over 1,000,000 alerts. I ran a Perl script that broke each day into smaller files of equal size then ran each of those files through SnortSnarf. Finally, I exported the data into an Excel spreadsheet to arrive at daily and alert totals. I believe it was the trends were ultimately more clear by handling the data in this manner.

I noted in several analyses that MY.NET had to be replaced. However, in the logs I downloaded, MY.NET had already been normalized.

The scan logs and out of spec logs were easily concatenated into single files, using the following command on a Solaris host:

```
$ cat scan-0208* > scans-all
```

Upon the realization that the payloads were missing, I grepped out the headers in each cat'd file to simplify the analysis, using the following command:

```
$ grep '\->' scans-headers
```

I imported the header files to the Windows host for ease of reference, opening

each WordPad to retain the formatting. I then spent several days manually reviewing the headers, making notes as I went through, which resulted in the top talkers and infected hosts lists. During the preparation of the formal analysis, I used Cygwin and Wingrep to apply *nix utility capabilities to arrive at service totals, individual host totals, and arriving at the severity levels.

Finally, I made frequent use of Google as a research tool.

References

Microsoft TechNet, "Microsoft Security Bulletin MS01-020," 29 March 2001
URL:
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-020.asp (10 October 2002)

Microsoft TechNet, "Microsoft Security Bulletin MS01-044," 15 August 2001
URL:
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-044.asp (10 October 2002)

Danyliw, Roman; Dougherty, Chad; Householder, Allen; Ruefle, Robin, "CERT Advisory CA-2001-26 Nimda Worm," 18 September 2001, rev. 25 September 2001
URL: http://www.cert.org/advisories/CA-2001-26.html (14 October 2002)

Author unknown, "Nimda," 20 September 2001.
URL:
http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci770982,00.html (14 October 2002)

Mooney, Brian, "Re: Wave of Nimda-like Hits This Morning?" 26 February 2002
URL: http://www.der-keiler.de/Mailing-Lists/securityfocus/incidents/2002-02/0212.html (14 October 2002)

Hernan, Shawn; Rain Forest Puppy, "Vulnerability Note VU#111677, Microsoft IIS 4.0/5.0 Vulnerable to Directory Traversal via Extended Unicode in URL (MS00-078)," 20 November 2000, rev. 18 September 2001
URL: http://www.kb.cert.org/vuls/id/111677 (10 October 2002)

Microsoft TechNet, "Microsoft Security Bulletin MS00-057," 10 August 2000.
URL:
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms00-057.asp (14 October 20020

CIAC Information Bulletin, "L-098: Microsoft Index Server ISAPI Extension Buffer Overflow," 19 June 2001.
URL: http://www.ciac.org/ciac/bulletins/l-098.shtml (14 October 2002)

Internet Storm Center "Current Status" banner, updated daily.
URL: http://isc.incidents.org (20 October 2002)

Microsoft TechNet, "Microsoft Security Bulletin MS01-033," 18 June 2001.
URL:
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp (14 October 2002)

Elliott, Bree. "GIAC Certification Practical, SANS Online Curriculum, v. 2.9." 2001.
URL: http://www.giac.org./practical/Bree_Elliott_GCIA.doc. Analyze This, page 35 (16 October 2002)

Arana, Gail, "Yale Limits Use of Kazaa, Other Clients," 15 October 2002.
URL: http://www.yaledailynews.com/article.asp?AID=20150 (16 October 2002).

Borland, John, "Hollywood Chases Down Campus Pirates," 10 October 2002.
URL: http://news.com.com/2100-1023-961637.html (16 October 2002)