Global Information Assurance Certification Paper

JIM BECHER

GCIA PRACTICAL V3.3

ASSIGNMENT 1: DESCRIBE THE STATE OF INTRUSION DETECTION

HTTPVER, GET /SUMTHIN!

SUMMARY
In July of 2002, Ben Laurie released an advisory that indicated serious vulnerabilities in OpenSSL. No exploit code was believed to exist at that time. Unix vendors released upgrades and patches to the vulnerabilities over the next week. In September, exploit code for several flavors of unix was developed and released. An additional piece of software was written (not by the author of the exploit), to aid in identifying vulnerable servers and calling the exploit code with the appropriate command-line arguments. This additional piece of software, httpver, generated odd log entries in webservers. A fact that was noticed in October of 2002. Httpver took advantage of the default Apache configuration which discloses operating system and web server version information through the Apache directives of ServerTokens and ServerSignature. Applying the patches, or upgrading, in a reasonable time would address this vulnerability. Another method of defeating httpver, is to change the default settings of the ServerTokens and ServerSignature directives in Apache.

We are going to walk through early reports of these strange log entries, how the httpver source code was discovered, acquiring and compiling the code, how the httpver code works, and the impacts and countermeasures. Packet traces are provided to illustrate one of the defensive mechanisms.

EARLY REPORTS
Back in October, 2002, we started seeing reports of log entries in webservers that were requests for "/sumthin". The earliest reference I found in a pubic forum or mailing list was an e-mail from jmaywood1975@hushmail.com dated 17 October, 2002. The e-mail can be found in the archives at SecurityFocus at the URL http://www.securityfocus.com/archive/75/295738/2002-10-18/2002-10-24/2. The post was made to the incidents list on SecurityFocus. The webserver was returning a 404, page not found error message. In addition, the logs posted in this initial e-mail also included log entries that were indicating problems with SSL connections from the same host at the same time. The earliest log entry that contained the "GET /sumthin" was from 10 October 2002, from IP address 205.150.215.204. Interestingly though, a couple of the ssl error messages involving this host were from 1 October 2002.

Several posters quickly affirmed that they were seeing the same things, but none of them indicated that they had logs prior to October 10th. One of the responders (zeno@cgisecurity.net) suggested that it might be a mechanism for banner grabbing. This turns out to be the case, but not the whole story.

HACKED MACHINE, WE FOUND "SUMTHIN"
On February 26, 2003, Philipp Hug forwarded an e-mail (http://www.securityfocus.com/archive/75/313283) to the incidents mailing list at SecurityFocus with the httpver.c source code. The code was retrieved from a machine

that had been compromised. In the e-mail from the victim administrator, he indicates that the "GET /sumthin"s are indeed a probe, but that the compromise is due to a bug in SSL – but not specific.

The following day, D.C. van Moolenbroek suspects that the SSL bug is the "openssl-too-open" mod_ssl exploit by Solar Eclipse (solareclipse@phreedom.org), from September of 2002.

It is interesting to note that the timeframe from detecting the activity to the httpver code being posted was in excess of 4 months. As I mentioned before, there were many people seeing this traffic from various sources. The httpver code was clearly being used from geographically disperse areas.

ACQUIRING AND COMPILING

I acquired the httpver.c source code from Philipp's post. I have a Redhat 8.0 machine that in my lab that we use for purposes such as these. The machine is secured, and resides on an internal segment.

The source code does not indicate the author, or attribute the code to a particular group. It does contain a couple of misspelled words, and a couple of words that appear to be Romanian.

Compiling this piece of code consists of a simple "gcc –o httpver httpver.c". It requires normal system functions and libraries.

Mr. Van Moolenbroek indicated that the "openssl-too-open" exploit code could be found at PacketStorm Security website. It is located at http://packetstormsecurity.nl/filedesc/openssl-too-open.tar.html, dated September 17th, 2002. From the description for openssl-too-open at PacketStormSecurity's website: "*OpenSSL v0.9.6d and below remote exploit for Apache/mod_ssl servers which takes advantage of the KEY_ARG overflow. Tested against most major Linux distributions. Gives a remote nobody shell on Apache and remote root on other servers*."

The file was retrieved from the PacketStormSecurity website, and unzipped and untar'd. Compiling the openssl-too-open exploit consists of a simple "make". A couple of warnings were encountered about the implicit declaration of the 'memcpy' and 'exit' functions in linux-x86.c.

HOW HTTPVER WORKS

The process flow of the httpver code is as follows:

- The code accepts a host and, optionally, a port as command-line arguments – if no port is provided, a default of 80 is assumed;
- It validates that either 1 or 2 command line arguments have been provided, otherwise it will display usage information;

- If 2 command line arguments are provided, it performs a sanity check on the port number;
- If a hostname is provided, the code resolves the hostname to an IP address, if resolution does not occur and error message is displayed and the code terminates;
- A socket is opened;
- A connection to the host is made, to the IP address and port number provided;
- If a connection to the IP address and port is unsuccessful, a connect error is logged and the code terminates;
- The default command, defined as "GET /sumthin HTTP/1.0/r/n/r/n" is sent;
- 99.99% of the time, this GET request will generate a 404 (Page Not Found) response, which often times contains webserver and host information;
- A buffer for the response is cleared, and data is received;
- Move data into a buffer called "buf" until all data has been received from the host;
- Close the connection to the host;
- Convert the entire response to lower-case;
- Locates the first occurrence of the string "\nserver:" in the data received;
- Extract the data from "\nserver:" to the first occurrence of "\r\n", if the length of this data is zero the code will log that it could not get the version and terminate;
- Log the IP address and server response to the logfile;
- A subroutine then checks to make sure it is an Apache 1.3.x server and returns an integer containing the subversion of 1.3, otherwise the subroutine will return a code (-1) to terminate;
- A subroutine then identifies the operating system (if it is returned in the 404 error message);
- Another subroutine then displays the "architecture", the operating system, and the version of Apache – the return code from this subroutine is the "architecture";
- If the operating system is known and the webserver version is vulnerable, the code then executes the following system call: "./openssl –a 0x%02x %s\n". It is executing a program named "openssl" in the current directory, and providing command-line arguments to it. In this case, it is passing it the "architecture" and the IP address of the server that is vulnerable to the openssl-too-open KEY_ARG vulnerability;
- If the operating system cannot be determined, but the webserver version is known to be vulnerable, the code will execute the openssl exploit multiple times, each time providing a different "architecture".

Here are packet traces to a machine that provides operating system and web server version information. The Apache server in question is an Apache 1.3.19 server, with a default installation, running on a Redhat 7.1 box. I have snipped the 3-way TCP handshake for brevity.

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

05/18-13:59:34.168596 192.168.1.4:34344 -> 192.168.1.3:80
TCP TTL:64 TOS:0x0 ID:14446 IpLen:20 DgmLen:77 DF
***AP*** Seq: 0x8498CA77  Ack: 0x5F293FD6  Win: 0x16D0  TcpLen: 32
```

```
TCP Options (3) => NOP NOP TS: 127514 95008
47 45 54 20 2F 73 75 6D 74 68 69 6E 20 48 54 54  GET /sumthin HTT
50 2F 31 2E 30 0D 0A 0D 0A                       P/1.0....

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

05/18-13:59:34.168596 192.168.1.3:80 -> 192.168.1.4:34344
TCP TTL:64 TOS:0x0 ID:19702 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x5F293FD6  Ack: 0x8498CA90  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 95008 127514

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

05/18-13:59:34.438596 192.168.1.3:80 -> 192.168.1.4:34344
TCP TTL:64 TOS:0x0 ID:19703 IpLen:20 DgmLen:561 DF
***AP*** Seq: 0x5F293FD6  Ack: 0x8498CA90  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 95035 127514
48 54 54 50 2F 31 2E 31 20 34 30 34 20 4E 6F 74  HTTP/1.1 404 Not
20 46 6F 75 6E 64 0D 0A 44 61 74 65 3A 20 54 75   Found..Date: Tu
65 2C 20 30 39 20 4A 61 6E 20 31 39 39 36 20 31  e, 09 Jan 1996 1
37 3A 33 31 3A 33 39 20 47 4D 54 0D 0A 53 65 72  7:31:39 GMT..Ser
76 65 72 3A 20 41 70 61 63 68 65 2F 31 2E 33 2E  ver: Apache/1.3.
31 39 20 28 55 6E 69 78 29 20 20 28 52 65 64 2D  19 (Unix)  (Red-
48 61 74 2F 4C 69 6E 75 78 29 20 6D 6F 64 5F 73  Hat/Linux) mod_s
73 6C 2F 32 2E 38 2E 31 20 4F 70 65 6E 53 53 4C  sl/2.8.1 OpenSSL
2F 30 2E 39 2E 36 20 44 41 56 2F 31 2E 30 2E 32  /0.9.6 DAV/1.0.2
20 50 48 50 2F 34 2E 30 2E 34 70 6C 31 20 6D 6F   PHP/4.0.4pl1 mo
64 5F 70 65 72 6C 2F 31 2E 32 34 5F 30 31 0D 0A  d_perl/1.24_01..
43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 63 6C 6F 73  Connection: clos
65 0D 0A 43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A  e..Content-Type:
20 74 65 78 74 2F 68 74 6D 6C 3B 20 63 68 61 72   text/html; char
73 65 74 3D 69 73 6F 2D 38 38 35 39 2D 31 0D 0A  set=iso-8859-1..
0D 0A 3C 21 44 4F 43 54 59 50 45 20 48 54 4D 4C  ..<!DOCTYPE HTML
20 50 55 42 4C 49 43 20 22 2D 2F 2F 49 45 54 46   PUBLIC "-//IETF
2F 2F 44 54 44 20 48 54 4D 4C 20 32 2E 30 2F 2F  //DTD HTML 2.0//
45 4E 22 3E 0A 3C 48 54 4D 4C 3E 3C 48 45 41 44  EN">.<HTML><HEAD
3E 0A 3C 54 49 54 4C 45 3E 34 30 34 20 4E 6F 74  >.<TITLE>404 Not
20 46 6F 75 6E 64 3C 2F 54 49 54 4C 45 3E 0A 3C   Found</TITLE>.<
2F 48 45 41 44 3E 3C 42 4F 44 59 3E 0A 3C 48 31  /HEAD><BODY>.<H1
3E 4E 6F 74 20 46 6F 75 6E 64 3C 2F 48 31 3E 0A  >Not Found</H1>.
54 68 65 20 72 65 71 75 65 73 74 65 64 20 55 52  The requested UR
4C 20 2F 73 75 6D 74 68 69 6E 20 77 61 73 20 6E  L /sumthin was n
6F 74 20 66 6F 75 6E 64 20 6F 6E 20 74 68 69 73  ot found on this
20 73 65 72 76 65 72 2E 3C 50 3E 0A 3C 48 52 3E   server.<P>.<HR>
0A 3C 41 44 44 52 45 53 53 3E 41 70 61 63 68 65  .<ADDRESS>Apache
2F 31 2E 33 2E 31 39 20 53 65 72 76 65 72 20 61  /1.3.19 Server a
```

```
74 20 31 32 37 2E 30 2E 30 2E 31 20 50 6F 72 74  t 127.0.0.1 Port
20 38 30 3C 2F 41 44 44 52 45 53 53 3E 0A 3C 2F   80</ADDRESS>.</
42 4F 44 59 3E 3C 2F 48 54 4D 4C 3E 0A           BODY></HTML>.

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

05/18-13:59:34.438596 192.168.1.4:34344 -> 192.168.1.3:80
TCP TTL:64 TOS:0x0 ID:14447 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x8498CA90  Ack: 0x5F2941D3  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 127541 95035

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

05/18-13:59:34.438596 192.168.1.4:34344 -> 192.168.1.3:80
TCP TTL:64 TOS:0x0 ID:14448 IpLen:20 DgmLen:52 DF
***A***F Seq: 0x8498CA90  Ack: 0x5F2941D3  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 127541 95035

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

05/18-13:59:34.478596 192.168.1.3:80 -> 192.168.1.4:34344
TCP TTL:64 TOS:0x0 ID:19704 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x5F2941D3  Ack: 0x8498CA91  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 95039 127541

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

Note the existence of the “Server: “ line, and that it contains the text “apache/1.3.19 (unix) (red-hat/linux) mod_ssl/2.8.1 openssl/0.9.6 dav/1.0.2 php/4.0.4pl1 mod_perl/1.24_01”. This information is the data that is parsed by the httpver code. In this case, it is determined that this server is vulnerable, and httpver calls openssl-too-open to exploit the server. I did not place the openssl-too-open exploit code in the current directory with httpver, so the exploit will not be launched against the server.

IMPACT AND COUNTERMEASURES

The impact of the httpver.c code is that it identifies vulnerable operating system and Apache version combinations. This information is collected, and then passed to an SSL exploit. The exploit allows for an attacker to remotely access a shell on the webserver. From there, an attacker would attempt to escalate their privileges to root, for a complete compromise.

Detection

A signature does not exist for the httpver code, perhaps due to the basic nature of the manner in which it gathers system and webserver information. If a signature is put in place looking for the “GET /sumthin HTTP/1.0” string, it would be incredibly easy for an attacker to modify the request to avoid detection. Detecting on the “Server: “ response

from the server I would think would lead to a significant number of false positives. I modified the httper code to request other URLs, and to use the HTTP HEAD request instead of a GET. Both allowed me to successfully identify a server as vulnerable. So clearly, focusing on the "GET /sumthin" request would only people who download the code and run it as-is.

A signature does exist that should detect the SSL exploit in the "misc.rules" file:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 443 (msg:"MISC
OpenSSL Worm traffic"; flow:to_server,established;  content:"TERM=xterm";
nocase; classtype:web-application-attack;
reference:url,www.cert.org/advisories/CA-2002-27.html; sid:1887; rev:2;)
```

While the signature does not appear to be built specifically for the openssl-too-open exploit, it should detect the "TERM=xterm" string from main.c.

```
$ grep xterm *
main.c:#define COMMAND1 "TERM=xterm; export TERM=xterm; exec bash -i\n"
```

Prevention

There are a couple of approaches to preventing httpver from successfully identifying a server as vulnerable: patch, disabling SSLv2 protocol negotiation, and configuring the server not to send operating system and webserver version information.

The short answer to mitigate being probed and exploited is to keep current on patches. Ben Laurie made the public announcement early on Jul 30, 2002. At that point, there were no known exploits. Several vendors (Debian, Trustix, Engarde, Gentoo, SuSE, Mandrake and Redhat) all made patch announcements, providing patches and/or upgraded versions to address the vulnerability, later that day. There were additional vendor announcements from FreeBSD, Slackware, Apple, and NetBSD over the next week.

The problem was that system administrators were not applying the patches. Eric Rescorla authored an interesting paper, entitled "Security Holes… Who cares?", on statistics and trends of system administrators patching/upgrading OpenSSL after the announcement of the KEY_ARG vulnerability. According to his paper, disabling SSLv2 protocol negotiation is an easy and effective countermeasure to the KEY_ARG vulnerability. Disabling SSLv2 protocol negotiation is as simple as a configuration directive and restarting the server.

In a post to the Full-Disclosure mailing list on September 17, 2002 – Solar Eclipse posted the exploit code. The post occurred more than a month after the vendor announcements listed above.

Another method of reducing the probability that someone will use these tools to probe and exploit your servers is to configure Apache not to include operating system and

Apache version information in responses to client requests. I edited the Apache configuration file, httpd.conf, and added the following two lines:

```
ServerToken prod
ServerSignature off
```

I then stopped and started Apache. Executing httpver again, generated the following traffic. Again, the 3-way TCP handshake was snipped for brevity:

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

05/18-14:25:48.388596 192.168.1.4:44721 -> 192.168.1.3:80
TCP TTL:64 TOS:0x0 ID:3883 IpLen:20 DgmLen:77 DF
***AP*** Seq: 0xE78D5622  Ack: 0xC1C98771  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 284936 252500
47 45 54 20 2F 73 75 6D 74 68 69 6E 20 48 54 54  GET /sumthin HTT
50 2F 31 2E 30 0D 0A 0D 0A                       P/1.0....

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

05/18-14:25:48.388596 192.168.1.3:80 -> 192.168.1.4:44721
TCP TTL:64 TOS:0x0 ID:23845 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xC1C98771  Ack: 0xE78D563B  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 252500 284936

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

05/18-14:25:48.398596 192.168.1.3:80 -> 192.168.1.4:44721
TCP TTL:64 TOS:0x0 ID:23846 IpLen:20 DgmLen:462 DF
***AP*** Seq: 0xC1C98771  Ack: 0xE78D563B  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 252500 284936
48 54 54 50 2F 31 2E 31 20 34 30 34 20 4E 6F 74  HTTP/1.1 404 Not
20 46 6F 75 6E 64 0D 0A 44 61 74 65 3A 20 54 75   Found..Date: Tu
65 2C 20 30 39 20 4A 61 6E 20 31 39 39 36 20 31  e, 09 Jan 1996 1
37 3A 35 37 3A 35 34 20 47 4D 54 0D 0A 53 65 72  7:57:54 GMT..Ser
76 65 72 3A 20 41 70 61 63 68 65 0D 0A 43 6F 6E  ver: Apache..Con
6E 65 63 74 69 6F 6E 3A 20 63 6C 6F 73 65 0D 0A  nection: close..
43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20 74 65  Content-Type: te
78 74 2F 68 74 6D 6C 3B 20 63 68 61 72 73 65 74  xt/html; charset
3D 69 73 6F 2D 38 38 35 39 2D 31 0D 0A 0D 0A 3C  =iso-8859-1....<
21 44 4F 43 54 59 50 45 20 48 54 4D 4C 20 50 55  !DOCTYPE HTML PU
42 4C 49 43 20 22 2D 2F 2F 49 45 54 46 2F 2F 44  BLIC "-//IETF//D
54 44 20 48 54 4D 4C 20 32 2E 30 2F 2F 45 4E 22  TD HTML 2.0//EN"
3E 0A 3C 48 54 4D 4C 3E 3C 48 45 41 44 3E 0A 3C  >.<HTML><HEAD>.<
54 49 54 4C 45 3E 34 30 34 20 4E 6F 74 20 46 6F  TITLE>404 Not Fo
75 6E 64 3C 2F 54 49 54 4C 45 3E 0A 3C 2F 48 45  und</TITLE>.</HE
```

```
41 44 3E 3C 42 4F 44 59 3E 0A 3C 48 31 3E 4E 6F  AD><BODY>.<H1>No
74 20 46 6F 75 6E 64 3C 2F 48 31 3E 0A 54 68 65  t Found</H1>.The
20 72 65 71 75 65 73 74 65 64 20 55 52 4C 20 2F   requested URL /
73 75 6D 74 68 69 6E 20 77 61 73 20 6E 6F 74 20  sumthin was not
66 6F 75 6E 64 20 6F 6E 20 74 68 69 73 20 73 65  found on this se
72 76 65 72 2E 3C 50 3E 0A 3C 48 52 3E 0A 3C 41  rver.<P>.<HR>.<A
44 44 52 45 53 53 3E 41 70 61 63 68 65 2F 31 2E  DDRESS>Apache/1.
33 2E 31 39 20 53 65 72 76 65 72 20 61 74 20 31  3.19 Server at 1
32 37 2E 30 2E 30 2E 31 20 50 6F 72 74 20 38 30  27.0.0.1 Port 80
3C 2F 41 44 44 52 45 53 53 3E 0A 3C 2F 42 4F 44  </ADDRESS>.</BOD
59 3E 3C 2F 48 54 4D 4C 3E 0A                    Y></HTML>.

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

05/18-14:25:48.398596 192.168.1.4:44721 -> 192.168.1.3:80
TCP TTL:64 TOS:0x0 ID:3884 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xE78D563B  Ack: 0xC1C9890B  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 284937 252500

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

05/18-14:25:48.398596 192.168.1.4:44721 -> 192.168.1.3:80
TCP TTL:64 TOS:0x0 ID:3885 IpLen:20 DgmLen:52 DF
***A***F Seq: 0xE78D563B  Ack: 0xC1C9890B  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 284937 252500

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

05/18-14:25:48.398596 192.168.1.3:80 -> 192.168.1.4:44721
TCP TTL:64 TOS:0x0 ID:23847 IpLen:20 DgmLen:52 DF
***A***F Seq: 0xC1C9890B  Ack: 0xE78D563C  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 252501 284937

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

05/18-14:25:48.398596 192.168.1.4:44721 -> 192.168.1.3:80
TCP TTL:64 TOS:0x0 ID:3886 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xE78D563C  Ack: 0xC1C9890C  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 284937 252501

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

Note the existence of the “Server: “ line, and that this time, it only contains the text “apache”. The httpver code is not able to determine the web server version, or the operating system that it is running on. Thus, the httpver code reports that the server is

not vulnerable. I realize that this information is available other ways, but this at least prevents to httpver tool from determining that the server is vulnerable.

REFERENCES

Hug, Philipp. “Re: More /sumthin.” SecurityFocus. 26 February 2003. URL: http://www.securityfocus.com/archive/75/313283

jmaywood1975@hushmail.com. “HTTP attack looking for /sumthin?” SecurityFocus. 17 October 2002. URL: http://www.securityfocus.com/archive/75/295738/2002-10-18/2002-10-24/2

Laurie, Ben. “OpenSSL Security Altert – Remote Buffer Overflows.” OpenSSL-Announce. 30 July 2002. URL: http://www.mail-archive.com/openssl-announce@openssl.org/msg00037.html

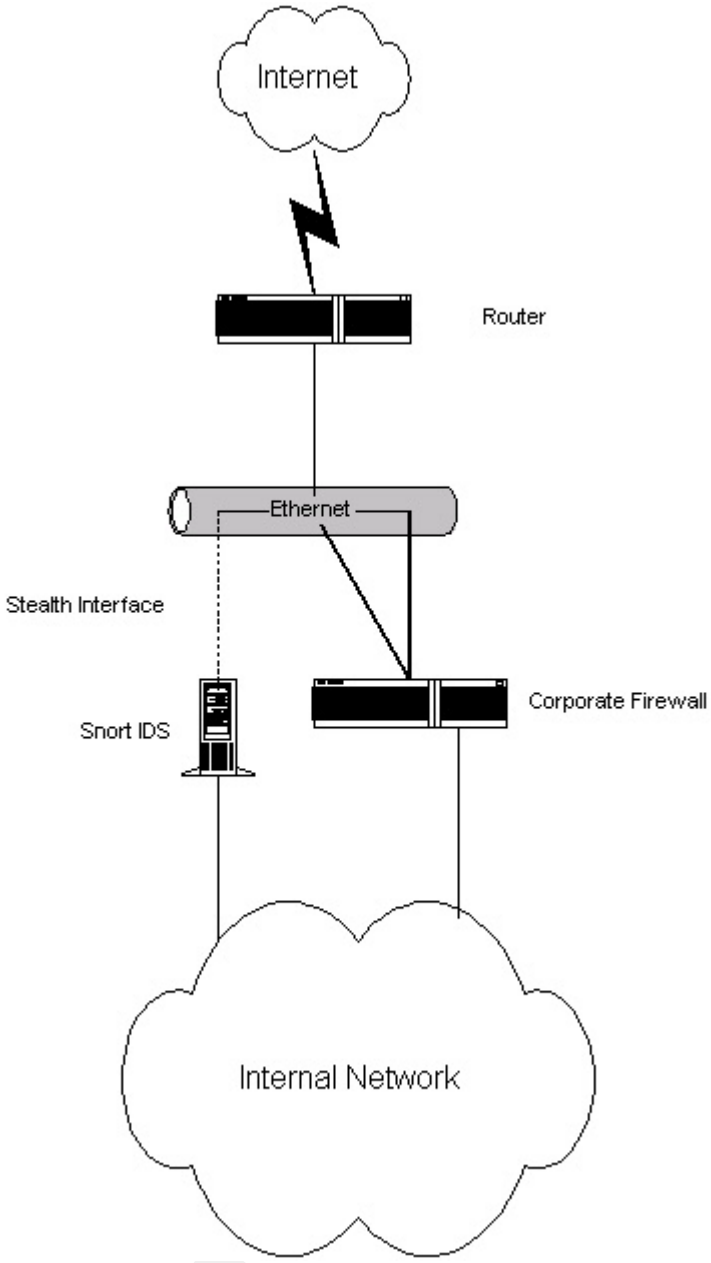
Packet Storm. “openssl-too-open.tar.gz." Packet Storm Security Tool Archive. 17 September 2002. URL: http://packetstormsecurity.nl/filedesc/openssl-too-open.tar.html

Rescorla, Eric. “Security Holes… Who cares?” URL: http://www.cgisecurity.com/lib/reports/slapper-report.pdf

ASSIGNMENT 2

DETECT #1 (SMTP HELO OVERFLOW ATTEMPT FROM 207.134.171.22)

1. *SOURCE OF TRACE.*
   The source of this trace is from my company's production network. The external segment consists of the ethernet interface of the Internet router, a Cisco 3550 ethernet switch, and the corporate firewalls. A spanning session was built on the Cisco 3550 ethernet switch. A network intrusion detection system is connected to the destination port of the spanning session. A Compaq DL-360, running Redhat Linux 8.0 is connected to the port on the Cisco 3550 switch where the spanning session is configured to send all traffic. The Compaq DL-360 has 2 ethernet interfaces configured – one has no IP address associated with it (stealth interface), and one with an internal IP address (management interface). The stealth interface on the Compaq DL-360 machine is connected to the external Cisco 3550 switch. The management interface on the Compaq DL-360 machine is connected to an internal Cisco 6509 switch.



2. *DETECT WAS GENERATED BY.*
   The detect was generated by Snort version 1.9.0 (Build 209). We run Snort in full packet capture mode, logging to binary files. The binary files are rotated hourly, at which time another instance of snort is run against the binary log file. The snort binary logs, and alerts, are logged locally to the machine. The rule which triggered the events is:

   alert tcp $EXTERNAL_NET any -> $SMTP_SERVERS 25 (msg:"SMTP HELO overflow attempt"; flow:to_server,established; content:"HELO "; offset:0; depth:5;

content:!"|0a|"; within:500; reference:cve,CVE-2000-0042;
reference:nessus,10324; classtype:attempted-admin; sid:1549; rev:9;)

This signature exists in the smtp.rules file.

alert
what snort is to do with the event when the signature triggers

tcp
traffic must be Transmission Control Protocol (TCP), protocol 6

$EXTERNAL_NET
the source of the traffic must match $EXTERNAL_NET (in our case is defined as "!$HOME_NET"), which means that it is anything that is not MY.HOME.NET.0/24

any
traffic can match any source port

$SMTP_SERVERS
the destination of the traffic must match $SMTP_SERVERS (in our case is defined as "$HOME_NET"), which means that it is any IP address in MY.HOME.NET.0/24

25
traffic must match the destination port of 25. Simple Mail Transfer Protocol (SMTP) uses TCP port 25 as its' default listening port.

msg: "SMTP HELO overflow attempt"
The name of the event

flow: to_server
indicates that the signature must match on traffic that is a client request to the server

established
this keyword indicates that the traffic must be part of a connection that has already successfully completed the TCP 3-way handshake of SYN, SYN-ACK, ACK.

content: "HELO "
snort is looking for this string of characters in the data stream, in this case the word HELO followed by a space

offset: 0
offset indicates the starting point within the payload of the packet to start looking for content

depth: 5
specifies how far into the payload of the packet to look for content

content: !"|0a|"
snort again is comparing the traffic to a string of characters, but in this case snort will only match on this content directive if the hex values of "0a" are not found. A hex "0a" is a linefeed.

within: 500
snort will match if a hex 0A is not found within the first 500 bytes of the payload

reference: cve, CVE-2000-0042 and reference: nessus,10324
References where analysts could go to get additional details on the vulnerabilities associated with this signature

classtype: attempted-admin
classtype is used for event prioritization and categorization, in this case "attempted-admin" means that triggering this signature indicates that someone is attempting to gain administrator privileges, which is classified as a high priority event

sid: 1549
a number used to uniquely track Snort rules

rev: 9
this is the revision of the signature

3. *PROBABILITY THE SOURCE ADDRESS WAS SPOOFED.*
The probability that the source address was spoofed is very low. The three-way handshake for the TCP connection completes successfully, and data is transmitted across the connection. For spoofing to occur, the true source of the packets would have to be in the path of traffic between the apparent source and destination of the connection.

4. *DESCRIPTION OF THE ATTACK.*
Beginning on the morning of April 16, 2003, hundreds of these events started rolling in per hour. The source address for all the alerts was the same. The destination indicated in the events was one of the corporate "bridgehead" Exchange servers that accepts mail from the Internet. Because we have snort configured to do full packet capture to a binary log file that is rotated hourly, I was able to go back and start

reviewing the entire set of connections. Here is one of the many connections that caused the Snort signature to trigger:

```
04/16-14:11:20.259592 207.134.171.22:39120 -> MY.HOME.NET.4:25
TCP TTL:47 TOS:0x0 ID:59535 IpLen:20 DgmLen:60 DF
******S* Seq: 0x4A933687  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 33349128 0 NOP WS: 0

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

04/16-14:11:20.259969 MY.HOME.NET.4:25 -> 207.134.171.22:39120
TCP TTL:127 TOS:0x0 ID:62386 IpLen:20 DgmLen:64 DF
***A**S* Seq: 0x92E82B  Ack: 0x4A933688  Win: 0xFAF0  TcpLen: 44
TCP Options (9) => MSS: 1380 NOP WS: 0 NOP NOP TS: 0 0 NOP NOP
SackOK

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

04/16-14:11:20.310485 207.134.171.22:39120 -> MY.HOME.NET.4:25
TCP TTL:47 TOS:0x0 ID:59536 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x4A933688  Ack: 0x92E82C  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 33349133 0

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

The above 3 packets show the completion of the TCP 3-way handshake.

```
04/16-14:11:20.311089 MY.HOME.NET.4:25 -> 207.134.171.22:39120
TCP TTL:127 TOS:0x0 ID:62395 IpLen:20 DgmLen:179 DF
***AP*** Seq: 0x92E82C  Ack: 0x4A933688  Win: 0xFAF0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 12544066 33349133
32 32 30 20 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  220 ************
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ****************
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ****************
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ****************
2A 2A 2A 2A 2A 2A 2A 30 2A 32 2A 2A 2A 2A 2A 2A  *******0*2******
32 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  2***************
2A 2A 2A 2A 2A 2A 2A 2A 2A 32 30 30 2A 2A 2A 2A  *********200****
2A 2A 2A 2A 2A 30 20 2A 30 2A 30 30 20 0D 0A     *****0 *0*00 ..

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

The Exchange server sends it's banner – it has been configured to mask its' banner.

```
04/16-14:11:20.363739 207.134.171.22:39120 -> MY.HOME.NET.4:25
```

```
TCP TTL:47 TOS:0x0 ID:59538 IpLen:20 DgmLen:57 DF
***AP*** Seq: 0x4A933688  Ack: 0x92E8AB  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 33349139 12544066
48 45 4C 4F 20                    HELO

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

The client sends a “HELO “ – but notice, there is no linefeed.

```
04/16-14:11:20.364050 207.134.171.22:39120 -> MY.HOME.NET.4:25
TCP TTL:47 TOS:0x0 ID:59537 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x4A933688  Ack: 0x92E8AB  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 33349139 12544066

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

04/16-14:11:20.613241 207.134.171.22:39120 -> MY.HOME.NET.4:25
TCP TTL:47 TOS:0x0 ID:59539 IpLen:20 DgmLen:57 DF
***AP*** Seq: 0x4A933688  Ack: 0x92E8AB  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 33349164 12544066
48 45 4C 4F 20                    HELO

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

04/16-14:11:21.117987 207.134.171.22:39120 -> MY.HOME.NET.4:25
TCP TTL:47 TOS:0x0 ID:59540 IpLen:20 DgmLen:57 DF
***AP*** Seq: 0x4A933688  Ack: 0x92E8AB  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 33349214 12544066
48 45 4C 4F 20                    HELO

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

04/16-14:11:22.112591 207.134.171.22:39120 -> MY.HOME.NET.4:25
TCP TTL:47 TOS:0x0 ID:59541 IpLen:20 DgmLen:57 DF
***AP*** Seq: 0x4A933688  Ack: 0x92E8AB  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 33349314 12544066
48 45 4C 4F 20                    HELO

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

04/16-14:11:28.116984 207.134.171.22:39120 -> MY.HOME.NET.4:25
TCP TTL:47 TOS:0x0 ID:59543 IpLen:20 DgmLen:57 DF
***AP*** Seq: 0x4A933688  Ack: 0x92E8AB  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 33349914 12544066
48 45 4C 4F 20                    HELO
```

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

04/16-14:11:36.158270 207.134.171.22:39120 -> MY.HOME.NET.4:25
TCP TTL:47 TOS:0x0 ID:59544 IpLen:20 DgmLen:57 DF
***AP*** Seq: 0x4A933688  Ack: 0x92E8AB  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 33350714 12544066
48 45 4C 4F 20                                   HELO

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

04/16-14:11:52.133512 207.134.171.22:39120 -> MY.HOME.NET.4:25
TCP TTL:47 TOS:0x0 ID:59545 IpLen:20 DgmLen:57 DF
***AP*** Seq: 0x4A933688  Ack: 0x92E8AB  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 33352314 12544066
48 45 4C 4F 20                                   HELO

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

The client continues to send “HELO “s at 0.25 seconds, 0.5 seconds, 1 second, 2 seconds, 4 seconds, 8 seconds, and 16 seconds. It was pointed out that the sequence numbers do not change, as they are retransmits. The packet at 2 seconds is not present, I can only assume that in this connection, that packet was dropped. The Exchange server is not responding.

```
04/16-14:12:20.411870 207.134.171.22:39120 -> MY.HOME.NET.4:25
TCP TTL:47 TOS:0x0 ID:59546 IpLen:20 DgmLen:68 DF
***AP**F Seq: 0x4A93368D  Ack: 0x92E8AB  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 33355143 12544066
74 62 6E 31 2E 62 72 64 6E 34 2E 63 6F 6D 0D 0A  tbn1.brdn4.com..

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

The client is apparently willing to give up, and sends a FIN-ACK one minute since the last packet from the Exchange server. In the payload of the FIN-ACK packet is a string that may be a hostname. The brdn4.com domain is also registered to Bluerockdove Inc.

```
04/16-14:12:20.412320 MY.HOME.NET.4:25 -> 207.134.171.22:39120
TCP TTL:127 TOS:0x0 ID:8462 IpLen:20 DgmLen:64 DF
***A**** Seq: 0x92E8AB  Ack: 0x4A933688  Win: 0xFAF0  TcpLen: 44
TCP Options (6) => NOP NOP TS: 12544668 33349139 NOP NOP Sack:
19091@13965

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

```
04/16-14:12:24.158651 207.134.171.22:39120 -> MY.HOME.NET.4:25
TCP TTL:47 TOS:0x0 ID:59547 IpLen:20 DgmLen:57 DF
***AP*** Seq: 0x4A933688  Ack: 0x92E8AB  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 33355514 12544668
48 45 4C 4F 20                                  HELO

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

04/16-14:21:33.167856 MY.HOME.NET.4:25 -> 207.134.171.22:39120
TCP TTL:127 TOS:0x0 ID:31927 IpLen:20 DgmLen:90 DF
***AP*** Seq: 0x92E8AB  Ack: 0x4A933688  Win: 0xFAF0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 12550196 33349139
34 35 31 20 54 69 6D 65 6F 75 74 20 77 61 69 74  451 Timeout wait
69 6E 67 20 66 6F 72 20 63 6C 69 65 6E 74 20 69  ing for client i
6E 70 75 74 0D 0A                                nput..

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

The Exchange server gives up on the client..

```
04/16-14:21:33.168269 MY.HOME.NET.4:25 -> 207.134.171.22:39120
TCP TTL:127 TOS:0x0 ID:31932 IpLen:20 DgmLen:52 DF
***A***F Seq: 0x92E8D1  Ack: 0x4A933688  Win: 0xFAF0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 12550196 33349139

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

…and sends a FIN-ACK

```
04/16-14:21:33.218717 207.134.171.22:39120 -> MY.HOME.NET.4:25
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:40 DF
*****R** Seq: 0x4A933688  Ack: 0x0  Win: 0x0  TcpLen: 20

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

04/16-14:21:33.219148 207.134.171.22:39120 -> MY.HOME.NET.4:25
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:40 DF
*****R** Seq: 0x4A933688  Ack: 0x0  Win: 0x0  TcpLen: 20

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

The client aborts the connection. It was pointed out that the client sends two RSTs, one for each of the packets the Exchange server sent.

According to RFC 821, "48 45 4C 4F 20" is not a valid response to a 220. There should be the client domain and a linefeed. I realize that RFC2821 has obsoleted RFC821, but RFC2821 does not really cover HELO.

It appears that the source is attempting to deliver mail, but is not compliant with the SMTP protocol as defined in RFC821/2821. I can't imagine a freeware or commercial client that is not compliant with a 1982 RFC. Perhaps someone is developing/testing a new tool for delivering spam.

Running the binary packet capture through p0f, indicates it is a Linux 2.4.x machine:

```
# p0f -s snort.log.1050519600 | grep "207.134.171.22"
p0f: passive os fingerprinting utility, version 1.8.2
(C) Michal Zalewski <lcamtuf@gis.net>, William Stearns
<wstearns@pobox.com>
p0f: file: '/etc/p0f.fp', 150 fprints, iface: 'eth0', rule: 'all'.
207.134.171.22 [18 hops]: Linux 2.4.2 - 2.4.14 (1)
```

It turns out that this was not an attempt to gain administrative privileges, so the event is a false positive in that respect. But it did bring to our attention that a spammer was attempting to deliver, or attempting to relay, spam.

```
$ whois -h whois.arin.net 207.134.171.22
[whois.arin.net]
TELUS Communications Inc. TELUS-207-134-0-0 (NET-207-134-0-0-1)
                              207.134.0.0 - 207.134.255.255
Telus Quebec TELUS-QC-207-134-160-0 (NET-207-134-160-0-1)
                              207.134.160.0 - 207.134.175.255
Blue Rock Dove BRD-8-207-134-170-0 (NET-207-134-171-0-1)
                              207.134.171.0 - 207.134.171.127
Blue Rock Dove BRD-8-207-134-170-0 (NET-207-134-171-0-2)
                              207.134.171.0 - 207.134.171.127
```

5. *ATTACK MECHANISM*

It turns out that this wasn't an attack. We were seeing the number of alerts per hour because the same connection was generating 6-9 alerts each. The number of connections per hour was not sufficient for a denial of service type attack.

6. *CORRELATIONS*

The references that are provided in the snort signature refer to the buffer overflow in the CSM Mail Server. Those references and correlations don't seem to be relevant. Instead, I queried the incidents mailing list. I got two responses off-list, and both indicate that they were seeing the same things on their networks. They provided no additional details or analysis.

I queried www.mynetwatchman.com, and there were two reports of this source address:

Most Recent Event Date/Time (UTC)	22 Apr 2003 18:39:11	16 Apr 2003 23:50:29
Agent Alias	Net2	kreator
Agent Type	win32	Perl
Log Type	BlackICE	Snort
Target Ip	207.51.x.x	161.53.x.x
# of Ips Targeted	1	1
IP Protocol	6	6
Target Port	25	25
Port/Issue Description	Simple Mail Transport Protocol (SMTP) SMTP port probe	Simple Mail Transport Protocol (SMTP) Simple Mail Transport Protocol (SMTP)
Source Port	55823	53506
Explanation	advICE | mNW Info	mNW Info
Event Count	1	1

A quick google search on “bluerockdove” and “spam” results in many, many hits. Also, the 207.134.171.0/24 network (SBL6884) is listed on the Spamhaus Block List (SBL) since the 14th of February 2003.

In addition, an “IP Info” report on dshield.org did not return any records.

7. *EVIDENCE OF ACTIVE TARGETING*
   There is evidence of active targeting. In this case, there is only one server open for port 25 on the network where this activity was detected. The firewall logs do not indicate a single dropped/rejected packet from the source IP address during the entire month of April.

8. *SEVERITY*
   Severity = (criticality + lethality) – (system countermeasures + network countermeasures)

Criticality=4

This is one of two corporate “bridgehead” Exchange servers.

Lethality=2

This was not an attack, it appears to have been an attempt to deliver or relay spam. If it would have been successful, we might have ended up on a open relays list, and we might not have been able to deliver mail to destinations that use those lists when accepting mail.

System Countermeasures=4

The server was completely patched, masked its’ banner, and does not allow relaying. To give a 5, I would like to have seen qmail, or similar, be used to accept mail from the Internet.

Network Countermeasures=4

The Exchange server is protected by routers performing ingress/egress filtering, redundant firewalls, and network IDS. It is acknowledged that our firewall and IDS would not have prevented the spam relaying from occurring if the Exchange server was misconfigured.

Severity = (4+2) – (4+4) = -2

9. *DEFENSIVE RECOMMENDATION*

Current defenses seem adequate, but one additional measure that could be taken would be to have qmail, or similar, accept all inbound mail from the Internet, and then forward it on to the internal Exchange servers. This reduces the risk of having a full-featured mail server exposed to the Internet. Small, secure, and efficient alternatives, such as qmail, should provide an additional level of security.

An improvement in the snort signature is probably needed. Matthew Callaway sent an e-mail to the snort-sigs list (http://sourceforge.net/mailarchive/forum.php?thread_id=2035256&forum_id=7141) containing an improved signature:

> From: snort-sigs-admin@lists.sourceforge.net [mailto:snort-sigs-admin@lists.sourceforge.net]On Behalf Of Matthew Callaway
> Sent: Tuesday, April 29, 2003 2:20 PM
> To: Ron Shuck
> Cc: snort-sigs@lists.sourceforge.net
> Subject: Re: [Snort-sigs] False Positive on SMTP HELO Overflow
>
> Here is a new version of this signature that works correctly:
>
> alert tcp $EXTERNAL_NET any -> $SMTP_SERVERS 25 (msg:"SMTP HELO overflow attempt"; flow:to_server,established; content:"HELO ";

offset:0; depth:5; content:!"|0a|"; within:500; content: "?"; offset: 499; regex; reference:cve,CVE-2000-0042; reference:nessus,10324; classtype:attempted-admin; sid:1549; rev:10;)

ie: "HELO " from byte 0 to 5, but no LF within 500 bytes, and at least one char at 500 bytes.

I then re-ran my binary captures past just this signature, and I received no alerts. I cannot say that this signature will trigger only on the buffer overflow attempt and that it does not false positive. But, I can say that it did not alert on the traffic that revision 9 of this signature had.

10. MULTIPLE CHOICE TEST QUESTION

Which one of the following RFCs discuss the Simple Mail Transfer Protocol?

A. RFC1149
B. RFC2821
C. RFC3514
D. It is not discussed in an RFC

Answer: B

The detect was posted to the intrusions@incidents.org mailing list on May 18, 2003. The text above includes changes that I have made based on the questions/comments from the list. Andrew Rucker Jones posed the following questions/comments to my detect:

>What does this (ARIN whois results) tell us?

Address space in North America is ultimately assigned and registered through ARIN (http://www.arin.net). ARIN maintains a whois service, that allows anyone to submit queries to their IP address database. The results from ARIN, in this case, tell us that the IP address 207.134.171.22 is part of the 207.134.0.0/16 Class B network assigned to TELUS Communications. It also has more granular entries indicating that a /20 CIDR chunk is assigned to TELUS Quebec. The contact information for bluerockdove.com has a physical street address in Montreal, Quebec.

>Can I reference those e-mails?

Andrew is referring to 2 replies I received off-list to my initial query about the network traffic I was seeing. As these e-mails were sent only to me, I feel that netiquette prevents me from providing any additional information.

>The scale is 1-5. If You need something to justify calling it "1" when it feels like it should be a "0", think >about the fact that spammers are using Your resources, which is a kind of attack. :)

Andrew is referring to the 0 score I provided for lethality. Andrew is correct. The score was changed based on his comments.

References

Callaway, Matthew. "Re: [Snort-sigs] False Positive on SMTP HELO Overflow." Snort-sigs Mailing List. 29 April 2003. URL: http://sourceforge.net/mailarchive/forum.php?thread_id=2035256&forum_id=7141

"Intrusion Reporting and Response." April 2003. URL: www.mynetwatchman.com

Klensin, J. "Simple Mail Transfer Protocol." Request for Comments 2821. April 2001. URL: ftp://ftp.rfc-editor.org/in-notes/rfc2821.txt

Postel, J. "Simple Mail Transfer Protocol." Request for Comments 821. 1 August 1982. URL: ftp://ftp.rfc-editor.org/in-notes/rfc821.txt

"The Spamhaus Project." Spamhaus Block List. April 2003. http://www.spamhaus.org/sbl/index.lasso

DETECT #2 (SCANNING FOR PROXIES)

1. *SOURCE OF TRACE.*
   The source of the detect was http://www.incidents.org/logs/Raw/2002.10.17. Binary logs are placed on the incidents website for use by students for their GCIA practical. These logs are sanitized, and only contain packets that violate the ruleset. Although not explicitly mentioned, it is assumed that the 170.129.x.x addresses are the obfuscated addresses. The addresses that are non-local to the organization that is providing the sanitized logs are assumed not to be obfuscated, as they will be used in correlating events with other services (i.e. dshield and mynetwatchman).

   It appears that the IDS was watching an untrusted, external segment. I am led to believe this due to the variety of scans/probes that the IDS saw. There are packets with a destination port of 0, and there are proxy (SOCKS, Squid, 8080/tcp) scans with only the SYN bit set. I would not expect that an organization would allow this type of traffic from the Internet to a DMZ or internal network. I would expect a filtering router or firewall to block this type of traffic.

   In addition, the Media Access Control (MAC) addresses also provide some credence to this assumption. A MAC address is compromised of a 6-hex-character OUI, and a device specific 6-hex-character string. All inbound traffic from non-170.129.x.x addresses have the source MAC address of 00:03:E3:D9:26:C0. All outbound traffic from 170.129.x.x address have the source MAC address of 0:0:C:4:B2:33. According to the IEEE (http://standards.ieee.org/regauth/oui/oui.txt), both the "00:03:E3" and the "00:00:0C" OUIs are registered to Cisco Systems. Cisco is a large manufacturer of routers and firewalls.

2. *DETECT WAS GENERATED BY.*
The detect was generated by Snort version 1.9.1 (Build 231). The rules which triggered the events are:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 1080 (msg:"SCAN SOCKS
Proxy attempt"; flags:S; reference:url,help.undernet.org/proxyscan/;
classtype:attempted-recon; sid:615; rev:3;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 3128 (msg:"SCAN Squid Proxy
attempt"; flags:S; classtype:attempted-recon; sid:618; rev:2;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 8080 (msg:"SCAN Proxy
\(8080\) attempt"; flags:S; classtype:attempted-recon; sid:620; rev:2;)
```

All three signatures exist in the "scan.rules" file. These three signatures match on TCP packets with only the SYN bit set, and that have destination ports of 1080, 3128, or 8080. In addition, the source of the packet must match $EXTERNAL_NET, and the destination of the packet must match $HOME_NET. If these signatures match network traffic, snort will alert with the description that is contained in the "msg" option. The classtype option for each of these signatures is an attempted recon, which means that someone is attempting to gather information. An attempted recon is classified as a medium priority event. The "sid" option is a unique tracking number for this signature within snort, and the "rev" option tracks the revision number of this particular signature.

In order to narrow the traffic and alerts down the specific host in question, I ran the command "snort –r 2002.10.17 –c /etc/snort/snort.conf –l ./2002.10.17-log/specific/ host 202.108.254.200". Here is a sample of the alerts that were generated for an example scan for a single host across all three ports:

```
[**] [1:620:2] SCAN Proxy (8080) attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/17-06:57:31.896507 202.108.254.200:8576 -> 170.129.38.209:8080
TCP TTL:46 TOS:0x0 ID:61101 IpLen:20 DgmLen:40
******S* Seq: 0x538A1F3F  Ack: 0x538A1F3F  Win: 0x400  TcpLen: 20

[**] [1:618:2] SCAN Squid Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/17-06:57:33.906507 202.108.254.200:55536 -> 170.129.38.209:3128
TCP TTL:46 TOS:0x0 ID:53371 IpLen:20 DgmLen:40
******S* Seq: 0x469C4525  Ack: 0x469C4525  Win: 0x400  TcpLen: 20

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/17-06:57:37.146507 202.108.254.200:31904 -> 170.129.38.209:1080
```

TCP TTL:46 TOS:0x0 ID:61585 IpLen:20 DgmLen:40
******S* Seq: 0x764EB76A Ack: 0x764EB76A Win: 0x400 TcpLen: 20
[Xref => url help.undernet.org/proxyscan/]

3. *PROBABILITY THE SOURCE ADDRESS WAS SPOOFED.*
While it is possible that the source address was spoofed, it probably is not in this case. The attacker is looking for open proxies. The only possible scenario where I could envision that these packets were spoofed, and still have value to the true attacker is if the true attacker was in the flow of packets to the spoofed source. In this scenario, the true attacker would see the responses from the hosts being scanned.

4 *DESCRIPTION OF THE ATTACK.*
In this case, the attacker sent SYN packets to ports 1080/tcp, 3128/tcp, and 8080/tcp. From the log files available, no other ports were scanned from this host, but 202.108.254.204 also ran some proxy scans through the IP space on the same day, but to different destinations. Given that the hosts are from the same organization and scanning for the same services, it is likely that these scans are coordinated. The 202.108.254.204 host has a different scan pattern than that of 202.108.254.200 – the .200 scans are close together on each host and are close together across multiple hosts. The 202.108.254.204 scans are slow across the ports and slow across the different hosts. Scans from both hosts do not have TCP options. If the scans would have been for 3128/tcp and 8080/tcp only (http://www.sans.org/resources/idfaq/ring_zero.php) – I would have considered that this might be a troll looking for RingZero-infected machined. However, since it included a scan for 1080/tcp, I believe the attacker was looking for open proxies. We have 221 SYN packets from 202.108.254.200 across 74 hosts in the 170.129.x.x range. Here is an example of a scan for a single host across all three ports:

=+=

11/17-06:57:31.896507 202.108.254.200:8576 -> 170.129.38.209:8080
TCP TTL:46 TOS:0x0 ID:61101 IpLen:20 DgmLen:40
******S* Seq: 0x538A1F3F Ack: 0x538A1F3F Win: 0x400 TcpLen: 20

=+=

11/17-06:57:33.906507 202.108.254.200:55536 -> 170.129.38.209:3128
TCP TTL:46 TOS:0x0 ID:53371 IpLen:20 DgmLen:40
******S* Seq: 0x469C4525 Ack: 0x469C4525 Win: 0x400 TcpLen: 20

=+=

11/17-06:57:37.146507 202.108.254.200:31904 -> 170.129.38.209:1080
TCP TTL:46 TOS:0x0 ID:61585 IpLen:20 DgmLen:40
******S* Seq: 0x764EB76A Ack: 0x764EB76A Win: 0x400 TcpLen: 20

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

$ whois -h whois.apnic.net 202.108.254.200
[whois.apnic.net]
% [whois.apnic.net node-2]
% How to use this server        http://www.apnic.net/db/
% Whois data copyright terms    http://www.apnic.net/db/dbcopyright.html

inetnum:      202.108.0.0 - 202.108.255.255
netname:      CHINANET-BJ
descr:        CHINANET Beijing province network
descr:        Data Communication Division
descr:        China Telecom
country:      CN
admin-c:      CH93-AP
tech-c:       SY21-AP
mnt-by:       MAINT-CHINANET
mnt-lower:    MAINT-CHINANET-BJ
changed:      hostmaster@ns.chinanet.cn.net 20000101
status:       ALLOCATED PORTABLE
source:       APNIC

person:       Chinanet Hostmaster
address:      No.31 ,jingrong street,beijing
address:      100032
country:      CN
phone:        +86-10-66027112
fax-no:       +86-10-66027334
e-mail:       hostmaster@ns.chinanet.cn.net
e-mail:       anti-spam@ns.chinanet.cn.net
nic-hdl:      CH93-AP
mnt-by:       MAINT-CHINANET
changed:      hostmaster@ns.chinanet.cn.net 20021016
source:       APNIC

person:       sun ying
address:      Beijing Telecommunication Administration
address:      TaiPingHu DongLi 18, Xicheng District
address:      Beijing 100031
country:      CN
phone:        +86-10-66198941
fax-no:       +86-10-68511003
e-mail:       suny@publicf.bta.net.cn
nic-hdl:      SY21-AP
mnt-by:       MAINT-CHINANET-BJ
```

changed: suny@publicf.bta.net.cn 19980824
source: APNIC

5. *ATTACK MECHANISM*

I ran the packets through p0f, to see if I could get an idea about the operating system of the source of the packets. The p0f tool reported these SYN packets as an nmap scan. I was curious as to why p0f was not able to identify the operating system, so I went to have a look at the packets themselves. I immediately noticed the lack of TCP options. Combined with the p0f results indicating an nmap scan, I decided to run some nmap scans of my own.

I ran a standard TCP portscan with nmap: "nmap –sT –p 1080,3128,8080 <IP address>" against a host under my control. At the same time, I had an instance of snort performing a binary packet capture. When I had snort display the packets of my first nmap scan, the TCP options were present.

Then I decided to run a Stealth SYN portscan with nmap: "nmap –sS –p 1080,3128,8080 <IP address>" against the same host under my control. Again, I had an instance of snort performing a binary packet capture. Bingo! This time the TCP options were not present, and running the packets from the second nmap scan through p0f yielded the same results as the packets from the detect.

There may be other tools, besides nmap, that could generate the same types of packets that we see in this detect.

6. *CORRELATIONS*

I used dshield.org and mynetwatchman.com to check to see if there is any other reported activity from this IP address. Dshield.org did not return any hits, but mynetwatchman,com did. There were four incidents reported to mynetwatchman.com since October of 2002 – incident number 10173264, 13845908, 17807917, and 24993599. In particular, incident 13845908 was of interested because it occurred on the same day as the scanning from this detect – November 17th, but it does not appear to contain any probes to 1080/tcp. It is unclear whether the entity that submitted this incident was monitoring for 1080/tcp. The other incidents did contain the inclusion of port 1080/tcp, and in some cases other additional ports. Given the timeframe, source IP address, and the ports scanned – I would say there is sufficient evidence to assume this detect trace and the incidents reported to mynetwatchman.com are related.

7. *EVIDENCE OF ACTIVE TARGETING*

There is no evidence of active targeting. There was no indication in the binary log file to indicate that these scans are in response to traffic directed at the target, or that there was additional host enumeration (ICMP pings for example) prior to the scans. This is simply someone scanning through blocks of IP addresses looking for

open proxies to use. In this case, the attacker scanned 74 hosts across the 170.129.x.x class B network.

8. *SEVERITY*

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

Criticality = 2

There is no evidence that these services are available, and being used by unauthorized users. However, depending on what segment the IDS is monitoring, this could indicate an improperly configured firewall or perimeter security device. If a system was accessible and configured as an open proxy, there are 3 concerns:

- the proxy could be used by outsiders, and consume network bandwidth;
- it may be possible to view internal, sensitive servers through an open proxy; and
- possible embarrassment to the company if the proxy was used to scan other systems or access indecent sites.

Lethality = 1

I don't believe that scans for open proxies themselves are cause for concern. I see many scans a day on the networks that I monitor. If an open proxy exists, it may indicate poor system and network administration.

System Countermeasures = 2

We are provided no information on the security of the hosts or network security mechanisms. Without additional information, it is difficult to determine an accurate severity based on the system countermeasures.

Network Countermeasures = 4

If my assumption that the IDS is watching the untrusted, external segment then I would rate the network countermeasures as a 4. If the organization has snort monitoring outside the firewall, I think it would be reasonable to assume that they are fairly security-savvy. The lack of response to the SYN packets could be that the firewall is configured to silently discard packets, or that the responses simply weren't included in the snort logs provided at the incidents.org website.

Severity = (2 + 1) – (2 + 4)
Severity = -3

9. *DEFENSIVE RECOMMENDATION*

If my assumption is correct that the IDS in this case is sitting in the external, untrusted segment outside the firewall, then my recommendations would be:

- Ensure these services (SOCKS, Squid, 8080/tcp) are not accessible from the Internet if not needed;

- If these services are needed, and must be accessible from the Internet, make sure they are secured appropriately so that they may not be used by unauthorized individuals;
- Report this scan to the netblock owner, and event aggregation services such as dshield.org or mynetwatchman.com.

If my assumption is not correct, and the IDS is monitoring a DMZ or internal network, then my recommendations would be:

- Review the configuration of the firewall, or perimeter security device, and block these services inbound if not needed;
- If these services are needed, and must be accessible from the Internet, make sure they are secured appropriately so that they may not be used by unauthorized individuals;
- Report this scan to the netblock owner, and event aggregation services such as dshield.org or mynetwatchman.com.

10. MULTIPLE CHOICE TEST QUESTION

A scan for open proxies on TCP ports 1080, 3128, and 8080 would be considered an attempted reconnaissance classtype. A default configuration of Snort would alert these with what priority?

A. Critical
B. High
C. Medium
D. Low

Answer: C

The detect was posted to the intrusions@incidents.org mailing list on May 18, 2003. The text above includes changes that I have made based on the questions/comments from the list. Andrew Rucker Jones posed the following questions/comments to my detect:

> You showed above that the addresses are from the Chinese telecom. Could this be a pool of dialup >addresses? In that case, Your assumption might not be safe. It might not hurt to mention the political >climate in China as support for this possibility: the government censors as much of the Internet as they >can, and Chinese citizens are constantly looking for ways around the censorship. Open proxies are one >way. Not that this can't be a completely normal open proxy scan that one would find coming from any >other country.

It is possible that this could be a pool of dialup addresses. Andrew brings up another reason to suspect that these are indeed scans looking for open proxies, as opposed to some other activity.

References

IEEE. April 2003. URL: http://standards.ieee.org/regauth/oui/oui.txt

"Intrusion Reporting and Response." April 2003. URL: www.mynetwatchman.com

Northcutt, Stephen. "What Was the Ring Zero Scan?" SANS Intrusion Detection FAQ. 11 October 1999. URL: http://www.sans.org/resources/idfaq/ring_zero.php

DETECT #3 (FTP CWD OVERFLOW – 7350WURM)

1. *SOURCE OF TRACE.*
   The source of the detect was http://www.incidents.org/logs/Raw/2002.10.17. Binary logs are placed on the incidents website for use by students for their GCIA practical. These logs are sanitized, and only contain packets that violate the ruleset. Although not explicitly mentioned, it is assumed that the 170.129.x.x addresses are the obfuscated addresses. The addresses that are non-local to the organization that is providing the sanitized logs are used to not be obfuscated, as they will be used in correlating events with other services (i.e. dshield and mynetwatchman).

   It appears that the IDS was watching an untrusted, external segment. I am led to believe this due to the variety of scans/probes that the IDS saw. There are packets with a destination port of 0, and there are proxy (SOCKS, Squid, 8080/tcp) scans with only the SYN bit set. I would not expect that an organization would allow this type of traffic from the Internet to a DMZ or internal network. I would expect a filtering router or firewall to block this type of traffic.

   In addition, the Media Access Control (MAC) addresses also provide some credence to this assumption. A MAC address is compromised of a 6-hex-character OUI, and a device specific 6-hex-character string. All inbound traffic from non-170.129.x.x addresses have the source MAC address of 00:03:E3:D9:26:C0. All outbound traffic from 170.129.x.x address have the source MAC address of 0:0:C:4:B2:33. According to the IEEE (http://standards.ieee.org/regauth/oui/oui.txt), both the "00:03:E3" and the "00:00:0C" OUIs are registered to Cisco Systems.

2. *DETECT WAS GENERATED BY.*
   The detect was generated by Snort version 1.9.1 (Build 231). The rule which triggered the event is:

   ```
   alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP wu-ftp bad file
   completion attempt {"; flow:to_server,established; content:"~"; content:"{";
   distance:1; reference:cve,CVE-2001-0550; reference:cve,CAN-2001-0886;
   reference:bugtraq,3581; classtype:misc-attack; sid:1378; rev:10;)
   ```

   This signature exists in the "ftp.rules" file. This signature matches on packets in established TCP connections from source addresses matching $EXTERNAL_NET and a destination address matching $HOME_NET. Also required for this rule to

match, the TCP connection must have a destination port of 21 (FTP command channel) and must contain the "~" and "{" characters in the payload flowing toward the FTP server.

If this signature matches network traffic, snort will alert with the description that is contained in the "msg" option. The classtype option for this signatures is "misc-attack", and is classified as a medium priority event. The "sid" option is a unique tracking number for this signature within snort, and the "rev" option tracks the revision number of this particular signature.

In order to narrow the traffic and alerts down the specific host in question, I ran the command "snort –r 2002.10.17 –c /etc/snort/snort.conf –l ./2002.10.17-log/specific3/ host 165.154.7.2". The alert that was generated is:

```
[**] [1:1378:10] FTP wu-ftp bad file completion attempt { [**]
[Classification: Misc Attack] [Priority: 2]
11/17-07:54:27.836507 165.154.7.2:1982 -> 170.129.50.4:21
TCP TTL:46 TOS:0x0 ID:35321 IpLen:20 DgmLen:68 DF
***AP*** Seq: 0x473AE04D  Ack: 0x719E0482  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 648881738 4580813
[Xref => bugtraq 3581][Xref => cve CAN-2001-0886][Xref => cve CVE-2001-
0550]
```

3. *PROBABILITY THE SOURCE ADDRESS WAS SPOOFED.*
I believe there is zero chance that the source address was spoofed. Successful exploitation of this vulnerability requires that the 3-way TCP handshake complete successfully, and that the attacker would have to supply a valid username/password pair prior to exploitation. In addition, I believe a published exploit (7350wurm.c) was used, and this tool does not have the capability to allow the user of the tool to provide a source IP address.

4. *DESCRIPTION OF THE ATTACK.*
Before we can discuss the vulnerability and the attack, a short introduction to "file globbing" is probably necessary. According to the CERT website (http://www.cert.org/advisories/CA-2001-07.html) that discusses this vulnerability, file globbing is "the process of expanding short-hand notation into complete file names". What this means is transforming "*.html" to mean all files that end in ".html". Or transforming "~homedir" into the actual path to the homedir directory (i.e. /export/home/homedir).

In this case, the vulnerability exploits a buffer overflow that exists when the globbing routines perform this transformation and provide larger results back to core routines of the FTP server. The buffer overflow allows for the execution or arbitrary commands by an attacker.

So, let's take a look at the traffic from 165.154.7.2:

```
$ snort -qdvr 2002.10.17 host 165.154.7.2
11/17-07:54:27.776507 165.154.7.2:1982 -> 170.129.50.4:21
TCP TTL:46 TOS:0x0 ID:35277 IpLen:20 DgmLen:560 DF
***AP*** Seq: 0x473ADE51  Ack: 0x719E0279  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 648881732 4580808
43 57 44 20 30 30 30 30 30 30 30 30 30 30 30 30  CWD 000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 F0 FC 40 31 07 08 98 5F 08 08 EB 0C  0000..@1..._....
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C 90 90 90 90 90 90 90 90 90 90 90 90  ................
31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1  1.C..tQ.-....P..
6A 04 58 89 C2 CD 80 EB 0E 31 DB F7 E3 FE CA 59  j.X......1.....Y
6A 03 58 CD 80 EB 05 E8 ED 0A CA 59 6A 03 58 CD  j.X........Yj.X.
80 EB 05 E8 ED FF FF FF FF FF FF 0A              ............
```

The shell code contained in this packet matches the source code for the 7350wurm.c (http://packetstormsecurity.nl/0205-exploits/7350wurm.c). Specifically:

```
/* x86/linux write/read/exec code (41 bytes)
 * does: 1. write (1, "\nsP\n", 4);
 *       2. read (0, ncode, 0xff);
 *       3. jmp ncode
```

```
 */
unsigned char   x86_wrx[] =
        "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

        "\x31\xdb\x43\xb8\x0b\x74\x51\x0b\x2d\x01\x01\x01"
        "\x01\x50\x89\xe1\x6a\x04\x58\x89\xc2\xcd\x80\xeb"
        "\x0e\x31\xdb\xf7\xe3\xfe\xca\x59\x6a\x03\x58\xcd"
        "\x80\xeb\x05\xe8\xed\xff\xff\xff";
```

In addition, I noticed the size of this packet is 560 bytes, with 32 bytes of TCP header. From the 7350wurm.c exploit code, there is a declaration of a variable "xpbuf" with appears to contain the payload of this packet. Here is the variable declaration:

```
unsigned char          xpbuf[512 + 16];
```

Notice that the size of the variable is 512+16, which is 528 bytes. 528 bytes of payload, plus the 32 bytes of TCP header yields 560 bytes.

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

11/17-07:54:27.836507 165.154.7.2:1982 -> 170.129.50.4:21
TCP TTL:46 TOS:0x0 ID:35321 IpLen:20 DgmLen:68 DF
***AP*** Seq: 0x473AE04D  Ack: 0x719E0482  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 648881738 4580813
43 57 44 20 7E 2F 7B 2E 2C 2E 2C 2E 2C 2E 7D 0A  CWD ~/{.,.,.,.}.
```

Again, this exact CWD (Change Working Directory) command matches the exploit code:

```
net_write (fd, "CWD ~/{.,.,.,.}\n");
```

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

11/17-07:54:28.186507 165.154.7.2:1982 -> 170.129.50.4:21
TCP TTL:46 TOS:0x0 ID:35599 IpLen:20 DgmLen:59 DF
***AP*** Seq: 0x473AE0B5  Ack: 0x719E05B9  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 648881774 4580850
43 57 44 20 7E 7B 0A                    CWD ~{.
```

And again:

```
net_write (fd, "CWD ~{\n");
```

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

$ whois -h whois.arin.net 165.154.7.2
```

[whois.arin.net]

OrgName: HookUp Communications
OrgID: HKUP
Address: 1075 North Service Road West, Suite 207
City: Oakville
StateProv: Ontario
PostalCode: L6M 2G2
Country: CA

NetRange: 165.154.0.0 - 165.154.255.255
CIDR: 165.154.0.0/16
NetName: HOOKUP-NET-4
NetHandle: NET-165-154-0-0-1
Parent: NET-165-0-0-0-0
NetType: Direct Assignment
NameServer: NS1.SISNA.COM
NameServer: NS2.SISNA.COM
Comment:
RegDate: 1993-06-18
Updated: 2002-12-02

TechHandle: BH922-ARIN
TechName: Harper, Benjamin
TechPhone: +1-801-924-0900
TechEmail: ipadmin@ikano.com

ARIN WHOIS database, last updated 2003-05-16 20:10
Enter ? for additional hints on searching ARIN's WHOIS database.

5. *ATTACK MECHANISM*

p0f gave no results for the 165.154.7.2 address… but manually looking at the characteristics of the packets – the source address seems to be a Linux 2.2 machine:

wwww:ttt:mmm:D:W:S:N:I:OS Description
#
wwww - window size
ttt - time to live
mmm - maximum segment size
D   - don't fragment flag (0=unset, 1=set)
W   - window scaling (-1=not present, other=value)
S   - sackOK flag (0=unset, 1=set)
N   - nop flag (0=unset, 1=set)
I   - packet size (-1 = irrevelant)
#

packets from 165.154.7.2 == 32120:64:???:1:?:?:1:

```
$ grep ^32120:64 /etc/p0f.fp
32120:64:1460:1:0:1:1:60:Linux 2.2.9 - 2.2.18
32120:64:1460:1:190:1:1:60:Linux 2.2.16
32120:64:1460:0:-1:0:0:44:Linux 2.0.38 (2)
32120:64:1460:1:101:1:1:60:Linux 2.2.15
32120:64:1460:0:-1:0:0:-1:Linux 2.0.33 (1)
32120:64:1460:0:0:1:1:60:Linux 2.2.19
32120:64:1460:1:9:1:1:60:Linux 2.2.x
32120:64:1460:1:100:1:1:60:Linux 2.2.14
```

6. *CORRELATIONS*

I consulted dshield.org and mynetwatchman.com to see if there were other reports of incidents of this type from the same source address. Dshield.org returned none matches. However, mynetwatchman.com returned 9 incidents, covering over 1300 events since July 2002. One of the incidents in particular, incident #13414043, contained reports of events on port 21 from the same day in this detect – Nov 17.

Incident Id	Source IP	Provider Domain	Agent Count	Event Count	Incident Status	ISP Resolution Comments
30089490	165.154.7.2	hookup.net	1	1	Closed	No Recent Activity
21453077	165.154.7.2	hookup.net	1	1	Closed	No Recent Activity
20392493	165.154.7.2	hookup.net	1	1	Closed	No Recent Activity
19467159	165.154.7.2	hookup.net	1	1	Closed	No Recent Activity
18315050	165.154.7.2	hookup.net	1	1	Closed	No Recent Activity
16405719	165.154.7.2	hookup.net	1	1	Closed	No Recent Activity
13414043	165.154.7.2	hookup.net	131	214	Closed	No Recent Activity
7845484	165.154.7.2	hookup.net	3	5	Closed	No Recent Activity
6442905	165.154.7.2	hookup.net	83	1108	Closed	No Recent Activity

As far as the vulnerability, snort provides a list of references in the alert. The references provided by snort are:

[Xref => bugtraq 3581]
which refers to Bugtraq BugID 3581, found at
http://www.securityfocus.com/bid/3581

[Xref => cve CAN-2001-0886]

which refers to Common Vulnerabilities and Exposures (CVE) Candidate 2001-0886, found at http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0886

[Xref => cve CVE-2001-0550]
which refers to CVE Entry 2001-0550, found at http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0550

In addition, CERT released a CERT Advisory for this particular vulnerability – CERT 2001-007, which can be found at http://www.cert.org/advisories/CA-2001-07.html.

7. *EVIDENCE OF ACTIVE TARGETING*

I would say that the possibility of active targeting is very likely. This was the only machine attacked for this vulnerability on this particular day. Also, on the day of the attack, no reconnaissance traffic is included in the binary snort logs. I also checked the three previous days looking for traffic from the same source IP address. It is possible that the traffic used to identify this machine as an FTP server simply did not get captured -- for instance if the 170.129.50.4 address was listed in DNS as an FTP server for this organization, or if there was a preceding scan for hosts with port 21 open.

8. *SEVERITY*

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

Criticality = 3

I think it is safe to assume that if an organization, outside of academia, is allowing FTP from the Internet, that it is serving a business/organizational purpose. And that disruption of this service would be business-impacting. It might not be as critical as DNS, SMTP, or WWW though.

Lethality = 5

This is a remote root exploit.

System Countermeasures = 2

We are provided no information on the security of the hosts or network security mechanisms. Without additional information, it is difficult to determine an accurate severity based on the system countermeasures.

Network Countermeasures = 4

If my assumption that the IDS is watching the untrusted, external segment then I would rate the network countermeasures as a 4. If the organization has snort monitoring outside the firewall, I think it would be reasonable to assume that they are fairly security-savvy. If even their network defenses were adequate, this vulnerability may exist over a service that the organization would allow across

their firewall (hopefully to a DMZ segment). It is acknowledged that a firewall or IDS would not have prevented this attack, but the IDS should (and did) alert on it.

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

9. *DEFENSIVE RECOMMENDATION*

My defensive recommendations would be to ensure that all systems that accept traffic from the Internet maintain a high level of security:

- Install only the minimum operating system and application components necessary to provide the services needed;
- That appropriate security configuration changes of the operating system are made – for instance, using the benchmarks from www.cisecurity.org
- That the application/service is secured, or use more secure alternatives. Do not allow anonymous FTP, enforce good password management (composition, expiration, lockouts, etc), etc in the case of FTP. If possible, use a more secure alternative such as scp, which is bundled with OpenSSH;
- Conduct a security assessment of the FTP server to see if it was vulnerable, and if an intrusion did occur.
- Install host-based IDS or file integrity checking software on the server. In the event that a zero-day exploit is every successfully used against this server, if the attacker makes any changes to critical system files, the host-based IDS or file integrity checking software should flag it.
- Join patch notification mailing lists, and quickly apply security patches.

From a network perspective, ensure that all inbound services from the Internet terminate on machines in a DMZ segment. This organization obviously has network IDS in place.

As far as incident response, I would recommend that the organization report this incident to the netblock owner as well as to 3rd party event aggregation services, such as dshield.org and mynetwatchman.com. In addition, the organization should document and track this incident in an internal trouble ticket system, if one exists. This will provide a record of the incident, the assessment of the server, the assessment of the success or failure of the attack, any recommendations made to the system/network administrators, etc.

10. *MULTIPLE CHOICE TEST QUESTION*

The Mitre CVE is an excellent resource for:

A. Repository of exploit code
B. Discussions surrounding how to code new exploits
C. Secure programming examples
D. A dictionary of vulnerabilities and exposures

Answer: D

References

Carnegie Mellon University CERT. "CERT® Advisory CA-2001-07 File Globbing Vulnerabilities in Various FTP Servers." 9 May 2001. URL: http://www.cert.org/advisories/CA-2001-07.html

IEEE. April 2003. URL: http://standards.ieee.org/regauth/oui/oui.txt

"Intrusion Reporting and Response." April 2003. URL: www.mynetwatchman.com

Stearns, Bill. "p0f." Bill Stearns' web site. 18 May 2003. URL: http://www.stearns.org/p0f/

TESO. "7350 wurm." Packet Storm Security Tools Archive. 2001. URL: http://packetstormsecurity.nl/0205-exploits/7350wurm.c

ASSIGNMENT #3: "ANALYZE THIS"

EXECUTIVE SUMMARY

As a part of the GCIA Practical, one of the assignments is to analyze 5 consecutive days of network security logs. For each day, there are 3 separate log files. All of the log files are generated by Snort. The purpose is for the student to demonstrate a strong understanding of the concepts covered in the GCIA course material, and to be able to apply them in a given scenario.

No other information regarding the other security measures/policies in place are provided, or information on network topology. Without a complete understanding of all the security measures or what networks contain critical computing infrastructure, it is not possible to provide a definitive assessment. However, based on the network traffic that we do have, we can provide some statistical information regarding the number and types of security events and alert them to compromised machines and other malicious traffic on their network. In addition, we can make some recommendations on how this organization can increase the security of their network.

Over the course of 5 days, from Sunday May 10th through Wednesday May 14th. During this time, snort detected thousands of alerts and scans. So many in fact, that we are only going to address the hosts involved in generating the most events and some other significant events. Many of these events are serious in nature, and require prompt attention. We will discuss other events that were detected, while they do not constitute a compromise, involve protocols/services that should be addressed from both a policy and technical standpoint.

Date	Alerts	Scans
May 10	262761	849285
May 11	455033	400869
May 12	144059	765336
May 13	159120	574013
May 14	190392	1211760
TOTAL	1211365	3801263

The data above clearly indicates that too many events exist for a reasonably-sized security team to review and resolve. Tuning of the IDS system, and incremental improvements in security should hopefully diminish these numbers to a more manageable amount.

The analysis below will provide details on machines that are compromised, or are being used by local users in malicious ways. One machine is scanning large portions of the Internet for machines tht have already been infected with a Trojan.

Another system is falsifying its' source address to attack a machine on the Internet. Still yet another machine is generating fragments of packets, most likely for some malicious purpose.

The analysis below will also provide details on how local users are misusing network bandwidth for the purpose of transmitting and receiving songs, that are most likely copyrighted. While this file sharing does not in itself pose a significant security risk, it does expose the users (and possibly the organization) to liability. This filesharing consumes valuable Internet bandwidth, that approved activities are also competeing for.

It is important to understand that there is risk associated with connecting an organization to the Internet. The analysis will show the shear magnitude at which the MY.NET systems and networks were scanned and probed by external organizations. Hopefully, this will emphasize the importance of good security policy and practices.

FILES ANALYZED

I selected files from the timeframe of May 10-May 14, 2003 to analyze. Each day consists of 3 different files: a scans file, an alerts file, and an OOS file. The scans file for each day contains information on portscans. The alerts file for each day contains information on the alerts that occurred – but only the bare minimum of information: date/time, the alert message from the signature, source IP address and port, and the destination IP address and port. The OOS, or Out-of-Spec, files contain malicious, or abnormal, traffic. The OOS files contain the packets that generated alerts. It also contains packets that snort has deemed to be abnormal, such as an invalid TCP flag combination or the lack of any TCP flags.

The alert files to be analyzed:

Filename	File Size
Alert.030510	23918429 bytes
Alert.030511	45332854 bytes
Alert.030512	13156809 bytes
Alert.030513	13787961 bytes
Alert.03514	17263604 bytes

The scan files to be analyzed:

Filename	File Size
Scans.030510	60169420 bytes

Scans.030511	27433835 bytes
Scans.030512	55843775 bytes
Scans.030513	41698343 bytes
Scans.030514	88248394 bytes

The OOS files to be analyzed:

Filename	File Size
OOS_Report_2003_05_11_20776	1443843 bytes
OOS_Report_2003_05_12_28902	1469443 bytes
OOS_Report_2003_05_13_31237	6379523 bytes
OOS_Report_2003_05_14_9396	1141763 bytes
OOS_Report_2003_05_15_16609	1372163 bytes

You will notice that the dates on the OOS files seem to indicate that they are from the May 11 – May 15 timeframe. Actually, the files contain the previous days OOS packets – thus the 2003_05_11 file actually contains data from May 10. Therefore, even though the filenames indicate that they cover May 11 – May 15, they actually cover the timeframe of interest, which is May 10 – May 14.

ANALYSIS

During the 5 day period, 49 different Snort signatures triggered from a pool of 1,211,365 total alerts. I think given the quantity of alerts and scans, it would be beneficial to focus on the major offenders in terms of alerts and portscans. Tod Beardsley, in his practical, focused on alerts of 10,000 events or more. This seems to be an adequate breakpoint at which to address the specific alerts individually.

Quantity	Snort Signature
323541	Incomplete Packet Fragments Discarded
199460	SMB Name Wildcard
47694	High port 65535 udp - possible Red Worm – traffic
23279	Tiny Fragments – Possible Hostile Activity
22991	spp_http_decode: IIS Unicode attack detected
17383	CS WEBSERVER – external web traffic
15919	High port 65535 tcp - possible Red Worm – traffic

11309	TFTP – Internal TCP connection to external tftp server

Incomplete Packet Fragments Discarded
In the course of normal network operation, it may be necessary to fragment larger packets to traverse links that have smaller maximum transmission unit (MTU). The MTU defines the maximum size of a packet that can traverse the communications link. When reaching the other end of the link, the fragments may be reassembled – this requires the remote end to hold the fragments in volatile memory until all the fragments are received and reconstructed to be passed along. Attackers have exploited this scenario to consume memory on the remote end, and cause a denial of service. This scenario was described in Ptacek and Newsham's paper entitled "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection" (http://www.snort.org/docs/idspaper/). It would be beneficial for devices to drop packet fragments that they have held for a period of time. Doug Kite, in his practical, indicates that these events are generated by a faulty NIC or that there is nefarious activity occurring. I am not inclined to believe this may be related to faulty hardware/NIC. During 4 of the 5 days, the alerts are very light throughout most of the day. Then for small timeframes in the evening, bursts of these alerts roll in. On May 11th, the activity was constant and strong all day. The internal IP address MY.NET.202.238 accounted for 99.98% of all these alerts, and of those alerts, 98% involved the external address 213.64.169.124 as the destination. The IP address 213.64.169.124 is registered to Telia. If this was a faulty NIC, I would have expected to see similar alerts to a multitude of destination addresses. In addition, the "scans" logfiles indicate a significant correlation to the amount of UDP traffic from MY.NET.202.238 to the 213.64.169.124 destination. The source ports associated with this traffic are mostly in the 1500-5000 range, however the destination ports appear to be random. The UDP protocol is used by streaming media applications, as it does not require the robustness, and thus the performance impact, of TCP. Perhaps these alerts coincide with an application/service that is running between these two hosts, for which the protocol is broken or not completely understood by snort and is being misinterpreted.

SMB Name Wildcard
SMB Name Wildcard is also known as NetBIOS Name Query, according to IDS177 at whitehats.com. The NetBIOS Name Query is covered as part of RFC 1002 – Protocol Standard For A NetBIOS Service On a TCP/UDP Transport. According to the whitehats.com summary, "Windows machines often exchange these queries as a part of the filesharing protocol to determine NetBIOS names when only IP addresses are known". While it is possible that a percentage of these events are related to the normal Windows NetBIOS name resolution process, we will describe a couple of hosts where this activity is related to scanning and information gathering. A SMB Name Wildcard request is a request for a list of any NetBIOS names known to the destination machine. An attacker could use these queries to determine the machine name, the domain the machine is in, and the username of the person logged in to the machine. This would be an information gathering/reconnaisance

type activity. All of these events, except 1, originated from outside of the MY.NET network. The alerts are fairly evenly distributed over the 5 day period:

alert.030510
48559 events from 5502 sources

alert.030511
34114 events from 5029 sources

alert.030512
34871 events from 6129 sources

alert.030513
39974 events from 6800 sources

alert.030514
41977 events from 6641 sources

There were 27,815 unique source IP addresses responsible for all 199,460 SMB Name Wildcard events (roughly about 7 events per source IP address). Only two source addresses was responsible for more than 500 SMB Name Wildcard events:

1 210.96.203.72 with 1,309 events to 1,309 unique destinations across 161 different Class C networks in the MY.NET network. All events were generated during roughly 2 hours on May 14th. The logs in the alert files are not exactly in time order. When sorting them in time order, it would appear that this source address scanned the entire MY.NET address space starting with MY.NET.1.0/24. The distribution across the 161 networks does not show any significant preference to any particular network – with the MY.NET.199.0/24 network having the highest number of events at 18. I generated a breakdown of SMB Name Wildcard events by destination network from this source address using the script smb-source1-dest-breakdown. It is not clear how the source selected the destination addresses – there is no indications in the scan logs for these 5 days.
2 218.29.219.1 with 617 events to 404 unique destinations across 4 different Class C networks in the MY.NET nework. All events were generated in under an hour on May 10th. The 404 events are fairly evenly distributed across the 4 networks (103, 93, 99, and 109). The source was using two different source ports, 137 and 1025, often to the same destination. The SMB Name Wildcard events occur with other probes from this source address for other Microsoft services. The scan logs indicate connections for 139/tcp, 445/tcp, and 137/udp during the same timeframe as the SMB Name Wildcard events.

My analysis does not lead me to believe that any one specific machine was singled out and targeted.

High port 65535 udp - possible Red Worm – traffic
The Red Worm, also known as Adore. According to F-Secure (http://www.f-secure.com/v-descs/adore.shtml), this worm infects Linux systems. The worm auto-propagates, attempting to infect other hosts by picking addresses at random, and exploiting known vulnerabilities in LPRng, rpc-statd, wu-ftpd, and BIND. If the worm is successful in exploiting one of these vulnerabilities, it will retrieve the worm code from a web server on the Internet. Once a machine has been infected, the worm installs a backdoor shell that listens on port 65535.

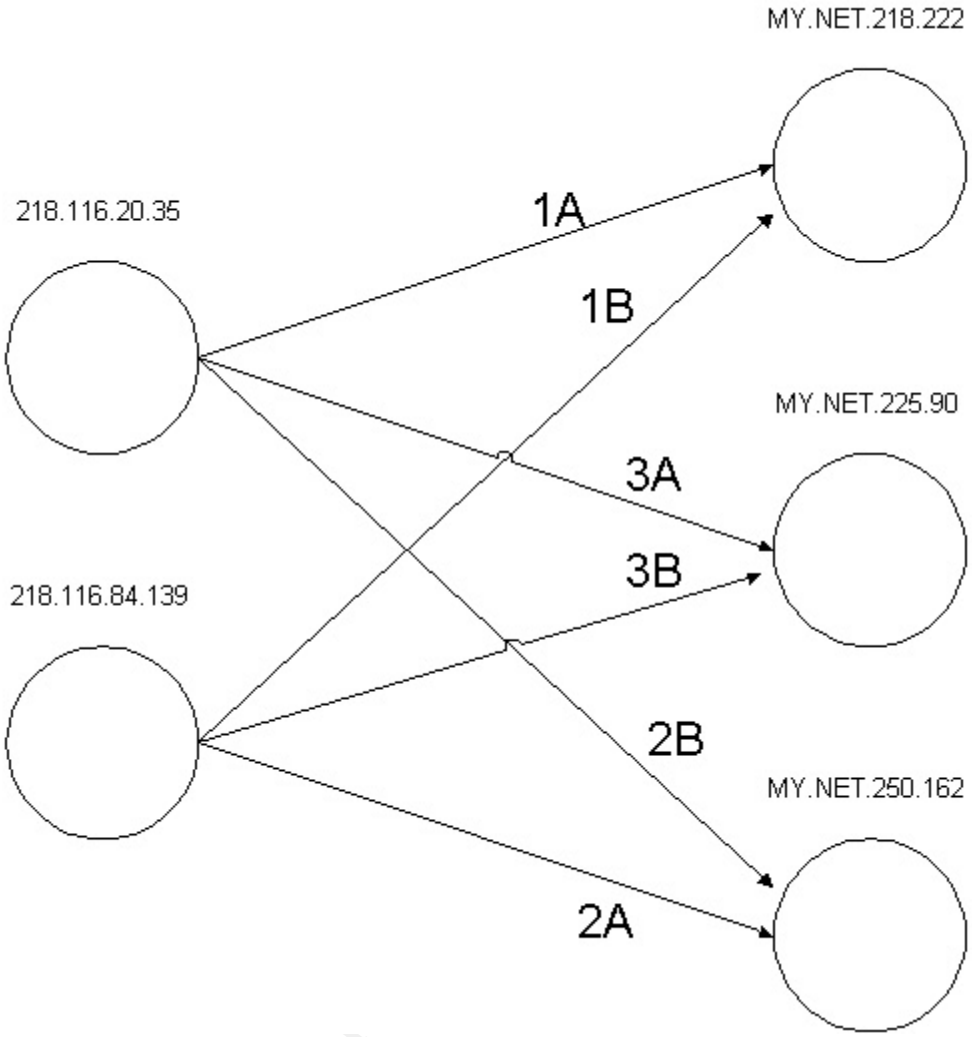
One host, MY.NET.201.58, was responsible for generating 40000+ of these alerts. Upon closer inspection, I believe that the MY.NET.201.58 host might be generating false positives. The scan logs indicate outgoing connections from port 65535 UDP to various hosts on the Internet with a destination ports of 5121 UDP and 13139 UDP. In addition, there are other connection events in the scan logs involving this host and with source and destination ports of 13139 UDP. Utilizing Google, I have discovered that the 13139 and 5121 ports are common, default ports for a game called Neverwinter – part of the GameSpy network. If you remove all the snort alerts for "High port 65535 udp – possible Red Worm – traffic" that involve ports 5121 and 5122, only 2657 alerts remain from the original 47,694 events. The connection between these game ports and 65535 UDP are not clear – google searches have not produced any clues. If packet traces are available, check to see if the traffic can be identified as a backdoor being accessed, or if the traffic appears to be related to a game. Otherwise, I still think it would be wise to investigate the MY.NET.201.58 host.

In the remaining 2657 alerts, if you break down the alerts by port number, obviously 65535 exists in all of them – but, port 6257 occurs in 2110 of the remaining 2657 alerts. UDP port 6257 is listed as WinMX in the portsdb.org database. WinMX is a Windows peer-to-peer filesharing program. The connection between the UDP port 6257 and UDP port 65535 are not clear – google searches have not produced any clues.

An interesting linkage diagram can be drawn from connections involving port 6257 and the 218.116.0.0/16 network on the Internet. The 218.116.0.0/16 address space is registered to SoftBank BB Corp in Japan. The activity involves two hosts on the 218.116.0.0/16 network and three hosts on the MY.NET network:

218.116.20.35
218.116.84.139
MY.NET.218.222
MY.NET.225.90
MY.NET.250.162

What is interesting is that it appears that there is some communication between the two hosts on the 218.116.0.0/16 network, whereby they are sharing/using information about hosts on the MY.NET network.



There are three pairs of UDP "connections" that indicate that there may be some level of communication between the 218.116.0.0 hosts:

1A At 2:42 AM on May 10th, snort alerted on traffic from 218.116.20.35:65535 to MY.NET.218.222:6257. There were no prior alerts, scan logs, or OOS packets involving these two hosts.

1B At 6:01 AM on May 10th, snort alerted on traffic from 218.116.84.139:65535 to MY.NET.218.222:6257. There were no prior alerts, scan logs, or OOS packets involving these two hosts. In fact, there was no activity from either of the 218.116. hosts between 2:42 AM and this alert at 6:01 AM.

2A At 4:52 PM on May 10th, snort alerted on traffic from 218.116.84.139:65535 to MY.NET.250.162:6257. There were no prior alerts, scan logs, or OOS packets involving these two hosts.

2B At 8:17 AM on May 11th, snort alerted on traffic from 218.116.20.35:65535 to MY.NET.250.162:6257. There were no prior alerts, scan logs, or OOS packets involving these two hosts.

3A At 12:39 PM on May 11th, snort alerted on traffic from 218.116.20.35:65535 to MY.NET.225.90:6257. There were no prior alerts, scan logs, or OOS packets involving these two hosts.

3B At 9:56 PM on May 11th, snort alerted on traffic from 218.116.84.139:65535 to MY.NET.225.90:6257. There were no prior alerts, scan logs, or OOS packets involving these two hosts.

For the second host to know the IP address and the port number, without scanning all hosts in the MY.NET network, seems impossible without the second host receiving information from another source. Another source could be the first host, or it could be from an application/service source – such as a worm or filesharing network.

If packet traces are available, check to see if the traffic can be identified as a backdoor being accessed, or if the traffic appears to be related to peer-to-peer filesharing. Otherwise, I still think it would be wise to investigate the three hosts MY.NET.218.222, MY.NET.225.90, and MY.NET.250.162.

Tiny Fragments – Possible Hostile Activity

According to the misc.rules file in my default Snort 1.9.1 installation, this rule detects IP packets with the More Fragments (MF) bit set, and length of less than 25 bytes. RFC1825 indicates that tiny fragment attacks are used to pass traffic through filters that are configured to disallow traffic. While the RFC refers to attacks with less than 8 bytes beyond the IP header, the snort rule flags on packet lengths of less than 25 bytes – or less than 5 bytes beyond the IP header (assuming no IP options have been set). The first four bytes beyond the IP header are the source port number (2 bytes) and the destination port number (2 bytes).

An internal machine, MY.NET.235.110, accounted for over 22,088 of these alerts spread over 849 destinations. The destination IP address seems to remain constant for lengths of times.

I can identify at least two possible causes behind the traffic that has been detected from MY.NET.235.110 – streaming media and actual attacks. Laurie Zirkle, in a post (http://www.incidents.org/archives/intrusions/msg02850.html) to the incidents.org intrusions mailing list, forwarded an e-mail that explained what she was seeing in her logs. The forwarded e-mail indicated that the source of the traffic she was seeing was a global traffic balancer. It was explained that the global traffic balancer probed her server in response to a request for streaming contents. The fact that the destination address remains constant for periods of time is then explainable if these alerts were indeed caused by streaming media.

The other explanation, and this seems more likely, is that the traffic is actually the MY.NET.235.110 host scanning/attacking other hosts. The number of destinations seems to high to all be related to streaming content. Additionally, if this was related to streaming media, I would expect to see more hosts on the MY.NET network

involved, and I would have expected the MY.NET.235.110 host to be the detination of the Tiny Fragments, not the source. The Tiny Fragment alerts are already in full swing in the first day's log files that I analyzed, so if this machine was compromised, it was done prior to May 10th. I recommend investigating this machine as soon as possible.

spp_http_decode: IIS Unicode attack detected
The use of unicode encoding to attempt to bypass tradition string-matching IDS signatures. A system is required for representing all characters (letters, punctuation, numbers) as numbers for computers to deal with them, referred to as encoding. Different encoding methods were used for languages with different character sets – which led to confusion and conflicts. A single encoding method was needed so that all character sets for all languages – unicode. Some attackers try to take advantage of some signature based IDS systems by encoding their attack strings in unicode, in hopes of bypassing the string-matching of the IDS system.

There were 22991 IIS Unicode "attacks" against 1104 different webservers, of which 281 are internal to MY.NET. It is safe to assume that these machines are indeed running webservers, because a TCP 3-way handshake would need to complete prior to the HTTP request which contained the unicode. While not strictly prohibited by RFC 793, data is not generally transmitted during the TCP 3-way handshake – therefore, for snort to trigger on an HTTP request containing unicode, the TCP 3-way shake must have already occurred. For the TCP 3-way handshake to have occurred, something must have been listening on port 80 of the destination machine.

There are only two MY.NET web servers that saw more than 50 IIS unicode attempts directed at them -- MY.NET.204.26 with 54 attempts, and MY.NET.222.166 with 615 attempts.

MY.NET.222.166 saw IIS Unicode attacks from 369 different sources. The most prolific source of the IIS Unicode alerts against this server was from 131.194.195.200, with 19 alerts. This address is registered to Trinity University in San Antonio, Texas. In addition to the 19 IIS Unicode alerts, there were also 3 CGI Null Byte attack alerts detected. In a post (http://archives.neohapsis.com/archives/snort/2000-11/0244.html) to the snort-users mailing list, Joe Stewart indicates that the CGI Null Byte alerts are raised when the http preprocessor detects the presence of "%00" in the http request. All the alerts were generated on the morning of May 13th, between 8-10 AM. There are no other alerts involving this source address, and there are no scan log or OOS packets that involve this host. I began to suspect that there may have been some active targeting going on, but the same pattern (8-15 IIS Unicode alerts, a couple of CGI Null Byte alerts, and nothing in the scan logs or OOS logs) held for the top 6 sources of IIS Unicode alerts against MY.NET.222.166.

MY.NET.204.26 saw IIS Unicode attacks from 3 different sources, with 1, 15, and 38 events. Again, the same pattern – mix of mostly IIS Unicode attack alerts with a

couple of CGI Null Byte alerts, with nothing in the scan or OOS log files – exists with MY.NET.204.26.

For these two hosts, I would investigate the webserver to see if it contains URLs that match the snort rule criteria for both the IIS Unicode attack and CGI Null Byte alerts. These specific rules do not exist in my default snort 1.9.1 installation, so I am unable to determine why network traffic is matching this signature.

Based on my personal experiences with a commercial IDS product, I have found a significant number of IIS Unicode "attacks" to be false positives.

CS WEBSERVER - external web traffic
Apparently, the organization is tracking external access to a web server in the CS department. This event seems to be specific to this organization.

High port 65535 tcp - possible Red Worm – traffic
The Red Worm, also known as Adore. According to F-Secure (http://www.f-secure.com/v-descs/adore.shtml), this worm infects Linux systems. The worm auto-propagates, attempting to infect other hosts by picking addresses at random, and exploiting 4 known vulnerabilities. If the worm is successful in exploiting one of these vulnerabilities, it will retrieve the worm code from a web server on the Internet. Once a machine has been infected, the worm installs a backdoor shell that listens on port 65535.

Of primary concern in this case would be any machines on MY.NET that we suspect are infected with the Red Worm, and thus only those alerts that involve traffic to or from a MY.NET host on TCP port 65535. In order to generate a list of machines that meet this criteria, I ran the red-worm-tcp-processing script. The results of the script include a count of the number of "High port 65535 tcp – Red Worm" alerts. The script generated 119 alerts across 18 hosts.

One host in particular, MY.NET.195.3, accounted for 79 of the 119 alerts. The 79 alerts are involve 73 distinct source IP addresses. I would definitely investigate the MY.NET.195.3 host, however, I think something other than Red Worm infection might be occurring. If someone discovered this as a Red Worm infected machine, I would expect to see fewer source addresses, but higher numbers of connections from those source addresses.

Another host, MY.NET.249.122, generated 5 alerts. The souce ports used to connect to MY.NET.249.122 are all 6348. The default port for the GNUtella filesharing protocol is 6346. This caused me to investigate the OOS logs for packets that include MY.NET.249.122. Sure enough, I found 125 SYN packets directed at MY.NET.249.122 for TCP port 6346. If packet traces are available, I would investigate these 5 connections to see whether or not these are associated with file sharing.

TFTP - Internal TCP connection to external tftp server
This snort signature is triggering on traffic on port 69/TCP between hosts on the MY.NET network and hosts that are not on the MY.NET network. Trivial File Transfer Protocol (TFTP) is a simple file transfer protocol that does not require authentication – only knowledge of the filename and location. One scenario where I know tftp is used often is in the storage and retrieval of configuration information for network devices (routers and switches).

We do not have the packet traces to confirm whether or not this is indeed TFTP traffic. However, if this is TFTP traffic, all the TFTP servers are external to the MY.NET network, and all the clients are on the MY.NET network. I was unable to correlate events prior to the TFTP connections that might indicate that the TFTP was occurring after some exploitation of the MY.NET machines. In some cases, there were no alerts involving the MY.NET machine prior to the TFTP alerts. In other cases, there was SMB Name Wildcard probes directed at the MY.NET machine prior to the TFTP alerts. In yet other cases, the MY.NET clients were generating IIS Unicode alerts while communicating to external web servers.

Of the 11300+ events, over 10500 events involve 43 tftp servers in the 64.12.x.x address space. This address space is registered to America Online (AOL). Another address space which is registered to America Online, 205.188.x.x, added another 9 tftp servers. Google searches did not turn up any clues as to the reason, or potential reasons, behind this traffic. Steve Lukacs noted similar alerts involving AOL destination addresses in the 64.12.x.x address space, but he did not shed any additional details.

There were 36 hosts total on the MY.NET network that were responsible for generating all the alerts. In particular, the 7 hosts below generated 98% of the alerts.

MY.NET.205.234	3568 alerts (all involve AOL address space)
MY.NET.240.10	3439 alerts (all involve AOL address space)
MY.NET.224.242	1800 alerts (all involve AOL address space)
MY.NET.242.34	943 alerts (all involve AOL address space)
MY.NET.194.91	630 alerts (all involve a 12.212.105.26 – AT&T)
MY.NET.223.114	467 alerts (all involve AOL address space)
MY.NET.235.114	399 alerts (all involve AOL address space)

The scan logs, the OOS logs, and the alert logs (other than the external tftp server alerts) did not turn up common items of interest involving these 7 hosts.

There were other significant alerts that, based on their severity, deserve to be mentioned.

Trojan Server Activity
In my default snort 1.9.1 installation, I do not have a signature that matches this alert message. Judging by the fact that all 3324 valid alerts include a host with a source or destination port of 27374, I will assume that is the extent of the signature logic. In 984 of the alerts, the source port is 27374 – which can occur during normal system operation. If the signature is just looking for SYN packets, these are more likely candidates for false positives than the others. If there is some content matching in the signature that is looking for content from the server to the client, then there is a higher degree of confidence that these are valid alerts.

A Bell Canada customer, 67.68.231.154, scanned 13 Class C netblocks in the MY.NET.0.0/16 network looking for hosts with port 27374 listening. This scanning generated 213 alerts, but I suspect that some traffic was missed or dropped along the way. The source port used by 67.68.231.154 increased with the same frequency as the destination IP address' last octect. Sorting the Trojan Server alerts involving this source address, gaps exist between the alerts – if the destination address skipped 3 IP addresses, then the source port also skipped 3 ports. I assume that the scan went sequentially through the address space, but we are only seeing a subset of the alerts.

Site Exec – Possible wu-ftpd exploit
On May 11th, an alert was raised indicating that a remote host might be attempting to exploit a vulnerability in an FTP server running on MY.NET.222.30. The reported source of the attack is 24.186.224.197, registered to Optimum Online. During the 5 days being analyzed, I see no other activity from this source address in the alert, scan, or OOS logs. The only other alert involving the MY.NET.222.30 host is an SMB Name Wildcard attempt several days later from a source address in Belgium. No entries appear in the scan or OOS logs for the MY.NET.222.30 host.

While the alert message does not exactly match any signature in my default snort 1.9.1 installation, it would appear that this alert is referring to SID 361 or SID 1971.

If it is referring to SID 361, a vulnerability was discovered in the way wu-ftpd was configured to handle the "site exec" command for users with a valid ftp account. If "site exec" is enabled on the server, and it is running a wu-ftpd version 2.4.1 or earlier, an attacker with a valid ftp account could gain root access remotely. While not specifically mentioned, I would assume anonymous ftp would be considered a valid ftp account.

If it is referring to SID 1971, a format string vulnerability was discovered with wu-ftpd in the routines that handle user input to the "site exec" command. Versions of wu-ftpd earlier than 2.6.2 are vulnerable. If "site exec" is enabled on the server, and it is running a vulnerable version of wu-ftpd, an attacker with a valid ftp account could gain root access remotely. Anonymous ftp is considered a valid ftp account. Multiple exploit scripts are available to exploit this vulnerability.

The machine is running an FTP server, because the user would have to be logged in to execute the “site exec” command. This would mean that the TCP 3-way handshake has already occurred, and that the user is logged in. If packet traces are available, the banner from the ftp server might be in the packet traces. This will give an indication what ftp server and what version of the ftp server are running. If the server is running wu-ftpd, ensure it is not vulnerable. If packet traces are not available, and if time and resources permit follow-up with the system administrator or owner of this host to be sure. Given that we have seen no other malicious traffic to or from this host since the “site exec” alert occurred, I suspect that no exploitation has occurred.

TCP SRC and DST outside network
At first, these alerts didn’t seem all that interesting or significant. During the 5 days being analyzed, 126 source addresses were responsible for generating the 805 alerts. I was fully expecting to see traffic from RFC 1918 address space and from the 169.254.x.x address space. When you remove the 23 source addresses from these reserved address spaces, the remaining 103 source addresses are fairly evenly distributed.

Looking at the destination addresses of the traffic that raised these alerts was a little more interesting. Of the 235 unique destination addresses, one host, 67.80.77.94, was the recipient in 485 of the 805 alerts. For these 485 alerts, there were 88 different source addresses used. Not only was the destination address the same, but the destination port was the same as well, TCP port 6112. It would appear that this might be related to the dtspcd vulnerability described in CERT Advisory 2002-01, located at http://www.cert.org/advisories/CA-2002-01.html. Given that the source of these packets are not on the MY.NET network, we might assume that someone is crafting these packets. Since this is TCP, the attacker is going to have a difficult time exploiting this using spoofed source addresses, unless the attacker is in the path of the return packets.

I would recommend implementing egress filtering (if it is not already being done) and tracking down the source of these packets. One method of tracking these back to the real source of the traffic is a little resource and time intensive. Start at the perimeter, and using a sniffer, capture traffic directed at the 67.80.77.94 address. Take a look at the packet captures, and note the source MAC address. This MAC address should tell you what device is putting these packets on the network you are currently monitoring. Once at that device, there may be several networks that feed that device that you may need to monitor to get the next hop back to the real source, but the process is the same.

External Scanners
Another analysis I conducted on the data is to determine would the most active, external scanners of the 130.85.x.x (aka MY.NET) were for each day, and cumulative for all 5 days.

May 10

IP Address	# Scans
195.222.17.185	12272
217.80.156.54	10358
12.208.65.249	7795
207.88.65.223	7014
211.167.135.133	6986
65.69.50.176	6255
211.90.196.118	6188
210.15.63.85	5121
213.47.191.249	4537
61.32.63.43	3812

May 11

IP Address	# Scans
210.202.229.175	4346
216.127.216.104	4082
61.54.97.241	3914
218.13.101.56	3871
218.61.105.129	3596
67.210.104.5	3282
61.170.217.99	2685
62.56.133.69	2629
4.47.43.245	2154
129.93.16.76	1926

May 12

IP Address	# Scans
210.6.114.193	6787
195.18.251.123	5468
213.77.159.197	3416
134.36.208.69	2271

208.206.10.122	1650
64.220.130.22	1168
206.55.70.34	851
68.210.178.210	475
68.37.242.151	300
207.21.208.83	212

May 13

IP Address	# Scans
128.125.49.73	7561
61.33.165.112	3294
81.91.66.73	2580
24.164.137.149	2377
218.18.83.219	1863
62.169.145.25	1748
211.189.77.86	1713
149.169.24.248	1645
81.50.68.53	1202
220.114.224.191	1139

May 14

IP Address	# Scans
220.71.31.138	6809
128.218.163.179	4120
155.135.17.1	3592
130.15.159.91	3589
139.130.198.160	3476
220.32.132.12	3041
12.235.52.207	2624
218.216.155.94	2578
217.230.59.205	2092
205.188.228.1	1960

5-day Cumulative

IP Address	# Scans
195.222.17.185	12272
217.80.156.54	10358
12.208.65.249	7795
128.125.49.73	7561
207.88.65.223	7093
211.167.135.133	6986
220.71.31.138	6809
210.6.114.193	6787
65.69.50.176	6255
211.90.196.118	6188

From these 6 tables, two interesting statistics stand out. First, 7 of the 10 most active, external scanners for the 5-day period were from Saturday, May 10th. This is contrary to what my organization experiences week after week. The second interesting statistic is that the all but one of the external scanners were contained to a single day. The one scanner (207.88.65.223) that was not contained to a single day, however, 98.8% of the portscan events from this host were on May 10th.

TOP TALKERS

Top Talkers (Alerters)

# Alerts	Source IP Address
330521	MY.NET.202.238
103096	MY.NET.196.193
25947	MY.NET.235.110
23522	MY.NET.201.58
12370	66.42.68.210
8131	MY.NET.227.198
6026	140.142.19.69
4745	MY.NET.195.99
4110	216.78.252.220

3607	213.77.159.197

Top Talkers (Scans)

# Scans	Source IP Address
2874935	130.85.196.193
94441	130.85.202.238
55696	130.85.227.198
22976	130.85.97.83
22241	130.85.251.142
14575	130.85.204.46
14549	130.85.249.178
14071	130.85.87.50
13675	130.85.236.178
13625	130.85.210.202

Top Talkers (OOS)

# Events	Source IP Address
18513	213.77.159.197
1041	66.117.21.91
802	209.123.49.137
404	148.63.137.221
392	210.253.206.180
160	200.167.111.33
156	209.47.197.14
155	81.218.92.11
152	213.186.35.9
151	209.47.197.12

** One of the top scanners did not show up in the scan logs. The source address 213.77.159.197 generated 18513 OOS packets by turning on the 2 reserved TCP flags ("1" and "2") with the SYN flag set. These packets were flagged as OOS, but did not show up in the scan logs. **

GATHER REGISTRATION INFORMATION ON 5 EXTERNAL SOURCES

I have selected the top 5 sources of external scans, from the Analysis section. The reason I selected these addresses for further investigation is so that an e-mail address for the netblock owner could be determined so that the scanning activity could be reported.

Host: 195.222.17.185

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

inetnum:      195.222.17.184 - 195.222.17.191
netname:      EE-VIRONE
descr:        Virone Reisiburoo AS - Kaubamaja 6
country:      EE
admin-c:      EV106-RIPE
tech-c:       EV106-RIPE
status:       ASSIGNED PA
mnt-by:       RIPE-NCC-NONE-MNT
changed:      cougar@data.ee 19970714
changed:      cougar@data.ee 19980508
source:       RIPE

route:        195.222.0.0/19
descr:        Data  Telecom, 195.222.0/19
origin:       AS3327
notify:       ripe@data.ee
mnt-by:       AS3327-MNT
changed:      tarmo@data.ee 19960819
source:       RIPE

person:       Enn Vilgo
address:      Virone Reisiburoo AS
address:      Kaubamaja 6
address:      Tallinn
address:      Estonia
phone:        +372 2 422 264
fax-no:       +372 2 421 056
nic-hdl:      EV106-RIPE
notify:       ripe@data.ee
changed:      cougar@data.ee 19970714
source:       RIPE
```

Host: 217.80.156.54

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

inetnum:      217.80.0.0 - 217.89.31.255
netname:      DTAG-DIAL14
descr:        Deutsche Telekom AG
country:      DE
admin-c:      DTIP
tech-c:       DTST
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:      ********************************************************
mnt-by:       DTAG-NIC
changed:      ripe.dtip@telekom.de 20001026
changed:      ripe.dtip@telekom.de 20030211
source:       RIPE

route:        217.80.0.0/12
descr:        Deutsche Telekom AG, Internet service provider
origin:       AS3320
mnt-by:       DTAG-RR
changed:      rv@NIC.DTAG.DE 20001027
source:       RIPE

person:       DTAG Global IP-Addressing
address:      Deutsche Telekom AG
address:      D-90449 Nuernberg
address:      Germany
phone:        +49 180 5334332
fax-no:       +49 180 5334252
e-mail:       ripe.dtip@telekom.de
nic-hdl:      DTIP
mnt-by:       DTAG-NIC
changed:      ripe.dtip@telekom.de 20030210
```

source: RIPE

person: Security Team
address: Deutsche Telekom AG
address: Germany
phone: +49 180 5334332
fax-no: +49 180 5334252
e-mail: abuse@t-ipnet.de
nic-hdl: DTST
mnt-by: DTAG-NIC
changed: abuse@t-ipnet.de 20030210
source: RIPE

Host: 12.208.65.249

[whois.arin.net]

OrgName: AT&T WorldNet Services
OrgID: ATTW
Address: 400 Interpace Parkway
City: Parsippany
StateProv: NJ
PostalCode: 07054
Country: US

NetRange: 12.0.0.0 - 12.255.255.255
CIDR: 12.0.0.0/8
NetName: ATT
NetHandle: NET-12-0-0-0-1
Parent:
NetType: Direct Allocation
NameServer: DBRU.BR.NS.ELS-GMS.ATT.NET
NameServer: DMTU.MT.NS.ELS-GMS.ATT.NET
NameServer: CBRU.BR.NS.ELS-GMS.ATT.NET
NameServer: CMTU.MT.NS.ELS-GMS.ATT.NET
Comment: For abuse issues contact abuse@att.net
RegDate: 1983-08-23
Updated: 2002-08-23

TechHandle: DK71-ARIN
TechName: Kostick, Deirdre
TechPhone: +1-919-319-8249
TechEmail: help@ip.att.net

OrgAbuseHandle: ATTAB-ARIN
OrgAbuseName: ATT Abuse
OrgAbusePhone: +1-919-319-8130
OrgAbuseEmail: abuse@att.net

OrgTechHandle: ICC-ARIN
OrgTechName: IP Customer Care
OrgTechPhone: +1-888-613-6330
OrgTechEmail: qhoang@att.com

OrgTechHandle: IPSWI-ARIN
OrgTechName: IP SWIP
OrgTechPhone: +1-888-613-6330
OrgTechEmail: swipid@nipaweb.vip.att.net

ARIN WHOIS database, last updated 2003-05-21 20:10
Enter ? for additional hints on searching ARIN's WHOIS database.

Host: 128.125.49.73

[whois.arin.net]

OrgName: University of Southern California
OrgID: USC-6
Address: University Computing Services
Address: University Park
City: Los Angeles
StateProv: CA
PostalCode: 90089-0251
Country: US

NetRange: 128.125.0.0 - 128.125.255.255
CIDR: 128.125.0.0/16
NetName: USCNET
NetHandle: NET-128-125-0-0-1
Parent: NET-128-0-0-0-0
NetType: Direct Assignment
NameServer: KAUS.USC.EDU
NameServer: USC.EDU
NameServer: SCF-FS.USC.EDU
NameServer: UUCP-GW-1.PA.DEC.COM
Comment:
RegDate: 1986-05-12
Updated: 1999-02-17

TechHandle: UNA2-ARIN
TechName: Network Administration, USCnet Network
TechPhone: +1-213-740-5555
TechEmail: netadmin@usc.edu

ARIN WHOIS database, last updated 2003-05-21 20:10
Enter ? for additional hints on searching ARIN's WHOIS database.

Host: 207.88.65.223

[whois.arin.net]

OrgName: XO Communications
OrgID: XOXO
Address: Corporate Headquarters
Address: 11111 Sunset Hills Road
City: Reston
StateProv: VA
PostalCode: 20190-5339
Country: US

NetRange: 207.88.0.0 - 207.88.255.255
CIDR: 207.88.0.0/16
NetName: XOXO-BLK-2
NetHandle: NET-207-88-0-0-1
Parent: NET-207-0-0-0-0
NetType: Direct Allocation
NameServer: NAMESERVER1.CONCENTRIC.NET
NameServer: NAMESERVER2.CONCENTRIC.NET
NameServer: NAMESERVER3.CONCENTRIC.NET
NameServer: NAMESERVER.CONCENTRIC.NET
Comment:
RegDate:
Updated: 2002-04-03

TechHandle: DIA-ORG-ARIN
TechName: DNS and IP ADMIN
TechPhone: +1-408-817-2800
TechEmail: hostmaster@concentric.net

OrgAbuseHandle: XCNV-ARIN
OrgAbuseName: XO Communications, Network Violations
OrgAbusePhone: +1-866-285-6208

OrgAbuseEmail: abuse@xo.com

OrgTechHandle: XCIA-ARIN
OrgTechName: XO Communications, IP Administrator
OrgTechPhone: +1-703-547-2000
OrgTechEmail: ipadmin@eng.xo.com

ARIN WHOIS database, last updated 2003-05-21 20:10
Enter ? for additional hints on searching ARIN's WHOIS database.

Bonus Host: 213.77.159.197
[whois.ripe.net]
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum: 213.77.159.128 - 213.77.159.255
netname: MIELEC-SDI
descr: TP S.A. SDI
descr: Mielec
country: PL
admin-c: WW199-RIPE
tech-c: SW3859-RIPE
status: ASSIGNED PA
mnt-by: TPNET
changed: tkielb@cst.tpsa.pl 20000913
source: RIPE

route: 213.77.0.0/16
descr: TPNET
descr: for abuse: abuse@tpnet.pl
origin: AS5617
mnt-by: AS5617-MNT
changed: nabn@tpnet.pl 20030228
source: RIPE

person: Wojciech Wozniak
address: ZT Rzeszow
address: POLAND
phone: +48 17 8525995
e-mail: wozniak@zt.rzeszow.tpsa.pl
nic-hdl: WW199-RIPE
mnt-by: TPNET
changed: tkielb@cst.tpsa.pl 19971104

changed: tkielb@cst.tpsa.pl 20020618
source: RIPE

person: Slawomir Wijowski
address: Zaklad Telekomunikacji Rzeszow
address: ul. Wyspianskiego 35
address: 35-001 Rzeszow
address: Poland
phone: +48 17 8525995
phone: +48 17 8525818
fax-no: +48 17 8525616
e-mail: wozniak@zt.rzeszow.tpsa.pl
nic-hdl: SW3859-RIPE
mnt-by: TPNET
changed: tkielb@cst.tpsa.pl 20000131
source: RIPE

ANY INSIGHTS INTO INTERNAL MACHINES SUCH AS COMPROMISE OR POSSIBLE DANGEROUS OR ANOMALOUS ACTIVITY.

Based on the sources of log entries in the scan log files, 130.85.196.193 is cause for concern. This host generated over 2.8 million portscan log entries while scanning large portions of the Internet for port 17300/TCP. According to www.portsdb.org, port 17300/tcp is associated with the Kuang2 Trojan. Information on this trojan can be obtained at www.glocksoft.com/trojan_list/Kuang2_the_virus.htm.

The scanning that was detected during these 5 days is corroborated by external event aggregation sources – dshield.org and mynetwatchman.com. On dshield.org, I performed an "IP Info" report on 130.85.196.193, and the results indicated that 1200+ attacks were reported on May 13th. My query to mynetwatchman.com resulted in incident ID 30282139. This incident ID covers the timeframe from May 7th through May 16th. During this timeframe, 16 agents reported over 1000 events.

This machine may either being directly controlled, or controlled through an IRC BotNet. I consider the later a possibility, due to the existence of a significant IRC connections each day from that host.

A machine on the MY.NET network appears to be spoofing packets from various source addresses, and appears to be attacking a single machine out on the Internet. Since the source addresses are forged, I cannot provide an address or machine name to track this activity back to.

Snort alerted on 40000+ events of the backdoor associated with Red Worm/Adore for the MY.NET.201.58 host. This signature should be reviewed for false positives, but in the meantime, the network traffic associated with these alerts should be reviewed. If it turns out that the network traffic is indeed Red Worm/Adore, than the MY.NET.201.58 host is infected, and very active.

From the OOS and scan logs, some users on the MY.NET network are engaging in peer-to-peer filesharing. The risks associated with peer-to-peer file sharing are both legal and technical. The industry groups that are chartered to protect artists have gotten more active in using the legal system to curtail peer-to-peer filesharing. From a technical perspective, peer-to-peer filesharing is an avenue for the introduction of malware, and can consume valuable Internet bandwidth.

A machine on the MY.NET network, specifically MY.NET.235.110, is generating traffic that is being alerted on by snort as containing "Tiny fragments". Tiny fragments can be used to bypass security filters, and cause instability on the destination host. MY.NET.235.110 is responsible for virtually all the 23279 "Tiny Fragment" alerts during the 5 day period of May 10th through May 14th.

I would contact America Online (AOL), and see if the TFTP traffic that was detected is expected bahvior or not.

DEFENSIVE RECOMMENDATIONS BASED UPON YOUR ANALYSIS.

There appears there might be some malicious use of systems on the MY.NET network and existence of worm and trojan activity, but given the user base I believe this is to be expected. The hope is that these incidents of worm and trojan activity are confined to user workstations and do not include the systems the organization relies upon for conducting operations. Anti-virus software, and frequent updates, should be a requirement to connecting a client system to the MY.NET network.

I would advocate tighter restrictions on inbound/outbound traffic to/from the organization, but I believe culturally this may be a challenge. At a minimum, I would recommend filtering NetBIOS in either direction at the perimeter, unless there are requirements for NetBIOS access across the perimeter. I would also recommend egress filtering at the perimeter. The organization should segregate critical computing infrastructure (DNS servers, SMTP servers, Web servers, etc) off from the rest of the network, and apply granular access control for access to those resources – if they haven't done so already.

I think increased user awareness is one area where this organization could realize some benefit. In parsing through the OOS data, I noticed a significant amount of peer2peer file sharing. The RIAA recently sent letters to university officials urging them to clamp down on file-swapping by students. I was unable to find a working link to the actual letter on the RIAA website, but the letter was covered in this news.com article -- http://news.com.com/2100-1023-961637.html. The threat of legal action from the RIAA, such as http://www.idg.net/ic_1281008_9691_1-5056.html, may be enough to deter some users. Using traffic shaping products, such as Packeteer's PacketShaper, give network administrators the ability to assign very little bandwidth to file sharing protocols. However, this may quickly escalate into an arms race with clever users, who merely alter the ports that their file-swapping activities use.

In addition, two of the top 5 snort alerts were SMB Name Wildcard and Red Worm activity. Providing information to users on simple steps they can take to secure their machines may help prevent them from becoming compromised or infected. Snort will still see probes, and attempts from external sources.

ANALYSIS PROCESS

Given the overwhelming amount of alerts and scans, my approach was first to identify the signatures or hosts that were generating an abnormally high number of events. I want to first identify, quantify, and understand the number of the events that were being generating while watching network traffic to and from MY.NET. The hope is that by addressing the hosts or signatures that were causing the majority of the events, that we could reduce the amount of events the security staff has to deal with down to a more manageable level.

The next step I took in my analysis process was to determine the major sources of scans from external sources. You may be asking why is this important, because there is probably very little that can be done with this information. These external machines are not under your control, and you might not ever identify the system/network administrator to determine the cause of the probe or scan. The benefit of gathering this information is in informing the netblock owner of the activity – in hopes of experiencing fewer probes/scans from the user or organization responsible. I have been unable to find any statistics or studies to support this theory, or how the use of 3rd party event aggregation services would be beneficial.

In order to categorize and quantify events and scans, I used a variety of shell scripts. The basis of the scripts are from Chris Calabrese's Practical, which can be found at http://www.giac.org/practical/Chris_Calabrese_GCIA.html.

The data files were kept separate to allow for easily looking at criteria from a day-by-day perspective, but it also allows us to aggregate the day for the 5-day period. A methodology that you will see in most of the scripts that were used to analyze the data.

SCRIPTS USED

I would like to acknowledge Chris Calabrese. In his practical, Chris documented the scripts that he used in generating his data. My choice was also to analyze the files using command-line tools, such as awk, grep, sort, and uniq. Using Chris' scripts as a basis was very beneficial.

Permission was obtained from Chris to use and modify the scripts from his practical.

top-scan-src-hosts

```
echo "Day-by-day breakdown"
for n in scan*
do
echo $n
cat $n | awk '$5 == "->" { print $4 }' | cut -d : -f 1 | sort | uniq -c | sort -rn | head
done
echo "Totals over 5-day period"
cat scan* | awk '$5 == "->" { print $4 }' | cut -d : -f 1 | sort | uniq -c | sort -rn | head
```

top-alerters

```
echo "Day-by-day breakdown"
for n in alert*
do
echo $n
grep '\[\*\*\]' $n | grep -v "End of portscan" | awk '/spp_portscan/ { print $7; next }    {
a=NF-2; print $a }' | cut -d : -f 1 | s
ort | uniq -c | sort -rn | head
done
echo "Total over 5 day period"
grep '\[\*\*\]' alert* | grep -v "End of portscan" | awk '/spp_portscan/ { print $7; next }
{ a=NF-2; print $a }' | cut -d : -f 1
 | sort | uniq -c | sort -rn | head
```

top-scan-src-hosts-ext

```
echo "Day-by-day breakdown"
for n in scan*
do
echo $n
cat $n | grep -v " 130.85." | awk '$5 == "->" { print $4 }' | cut -d : -f 1 | sort | uniq -c |
sort -rn | head
done
echo "Totals over 5-day period"
cat scan* | grep -v " 130.85." | awk '$5 == "->" { print $4 }' | cut -d : -f 1 | sort | uniq -c
| sort -rn | head
```

find-number-of-sources-of-smb-name-wildcard

```
for n in alert.03051*
do
grep "SMB Name Wildcard" $n | awk '{ print $7 }' | awk -F: '{ print $1 }' | sort | uniq -c
| wc -l
done
```

find-internal-machines-possibly-infected-with-redworm

```
grep "possible Red Worm - traffic" alert.03051* | awk '{ print $14; print $16 }' | grep
"65535" | sort | uniq -c | sort –n
```

find-all-hosts-involved-in-iis-unicode

```
grep "IIS Unicode attack detected" alert.03051* | awk '{ print $9; print $11 }' | sort |
uniq -c | sort -n >> iis-unicode
```

find-internal-servers-involved-in-iis-unicode

```
cat iis-unicode | grep ":80" | grep "MY.NET"
```

find-external-servers-involved-in-iis-unicode

```
cat iis-unicode | grep ":80" | grep -v "MY.NET"
```

find-source-of-tiny-fragments

```
grep "Tiny Fragments" alert.03051* | awk '{ print $10 }' | sort | uniq -c | sort –n
```

how-many-hosts-did-he-tiny-fragment

```
grep "Tiny Fragments" alert.03051* | grep "MY.NET.235.110 ->" | awk '{ print $12
}' | sort | uniq -c | sort -n | wc –l
```

top-talkers-alerts

```
grep '\[\*\*\]' alert* | grep -v "End of portscan" | awk '/spp_portscan/ { print $7; next }
{ a=NF-2; print $a }' | cut -d : -f 1 | sort | uniq -c | sort -rn | head
```

top-talkers-scans

```
cat scan* | awk '$5 == "->" { print $4 }' | cut -d : -f 1 | sort | uniq -c | sort -rn | head
```

find-smb-sources

```
for n in alert.03051*
do
    grep “SMB Name Wildcard” $n | awk ‘{ print $7 }’ | awk –F: ‘{ print $1 }’ >> smb-
    sources
done
```

find-smb-sources-count

```
cat smb-sources | sort | uniq –c | sort –n
```

smb-source1-dest-breakdown

```
for n in alert.03051*
do
    grep “210.96.203.72” $n | grep “SMB Name Wilcard” >> tmp-smb-source-raw-log
done
cat tmp-smb-source-raw-log | awk ‘{ print $9 }’ | awk –F: ‘{ print $1 }’ | awk –F. ‘{ print $3 }’ | sort –n | uniq –c
```

tinyfrag-235-110-raw

```
for n in alert.03051*
do
    grep “Tiny Fragments” $n | grep “MY.NET.235.110”>>235-110-tinyfrag-alerts-raw
done
```

tinyfrag-235-110-destination-breakdown

```
cat 235-110-tinyfrag-alerts-raw | awk ‘{ print $12 }’ | sort –n | uniq –c | sort –n >> 235-110-tinyfrag-alerts-destination-breakdown-results
```

red-worm-tcp-processing

```
for n in alert.03051*
do
    grep “High port 65535 tcp” $n >> red-worm-tcp-alerts
done
cat red-worm-tcp-alerts | awk ‘{ print $14; print $16 }’ >> red-worm-tcp-hosts
grep ^MY.NET red-worm-tcp-hosts | grep $65535 >> red-worm-tcp-hosts-mynet-65535
sort red-worm-tcp-hosts-mynet-65535 | uniq –c | sort –n
```

References

Beardsley, Tod. “Intrusion Detection and Analysis: Theory, Techniques, and Tools.” GCIA Practical. 8 May 2002. URL: www.giac.org/practical/Tod_Beardsley_GCIA.doc

Borland, John. “Hollywood Chases Down Campus Pirates.” Cnet. 10 October 2002. URL: http://news.com.com/2100-1023-961637.html

Calabrese, Chris. “SANS GCIA Intrusion Detection In Depth GCIA Practical Assignment.” GCIA Practical. December 2001. URL: http://www.giac.org/practical/Chris_Calabrese_GCIA.html.

Evers, Joris. “RIAA Sues Students for File-Swapping.” IDG News Service. 4 April 2003. URL: http://www.idg.net/ic_1281008_9691_1-5056.html,

F-Secure. “F-Secure Computer Virus Information Pages: Adore” April 2001. URL: http://www.f-secure.com/v-descs/adore.shtml

G-Lock Software. “Kuang2 the Virus.” Trojan Port List. May 1999. URL: http://www.glocksoft.com/trojan_list/Kuang2_the_virus.htm

“The Internet Ports Database.” 17 January 2002. URL: http://www.portsdb.org

Kite, Doug. “Intrusion Detection in Depth.” GCIA Practical. July 2002. URL: www.giac.org/practical/GCIA/Doug_Kite_GCIA.pdf

“Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications.” RFC 1002. March 1987. URL: ftp://ftp.rfc-editor.org/in-notes/rfc1002.txt

Reed, D; Traina, P; Ziemba, G. “Security Considerations for IP Fragment Filtering.” RFC 1858. October 1995. URL: ftp://ftp.rfc-editor.org/in-notes/rfc1858.txt

Vision, Max. “IDS177 NetBIOS-Name-Query.” arachNIDS: The Intrusion Event Database. 2001. URL: http://whitehats.com/info/IDS177

Zirkle, Laurie. “Explanation of Snort MISC Tiny Fragments from 211.13.231.126.” Intrusions Mailing List at Incidents.org. 22 January 2002. URL: http://www.incidents.org/archives/intrusions/msg02850.html

Last Updated: June 25th, 2018

Upcoming Training

SANS London July 2018	London, United Kingdom	Jul 02, 2018 - Jul 07, 2018	Live Event
SANSFIRE 2018	Washington, DC	Jul 14, 2018 - Jul 21, 2018	Live Event
Security Operations Summit & Training 2018	New Orleans, LA	Jul 30, 2018 - Aug 06, 2018	Live Event
SANS San Antonio 2018	San Antonio, TX	Aug 06, 2018 - Aug 11, 2018	Live Event
San Antonio 2018 - SEC503: Intrusion Detection In-Depth	San Antonio, TX	Aug 06, 2018 - Aug 11, 2018	vLive
Community SANS Columbia SEC503	Columbia, MD	Aug 13, 2018 - Aug 18, 2018	Community SANS
SANS Virginia Beach 2018	Virginia Beach, VA	Aug 20, 2018 - Aug 31, 2018	Live Event
SANS Tokyo Autumn 2018	Tokyo, Japan	Sep 03, 2018 - Sep 15, 2018	Live Event
SANS Amsterdam September 2018	Amsterdam, Netherlands	Sep 03, 2018 - Sep 08, 2018	Live Event
SANS London September 2018	London, United Kingdom	Sep 17, 2018 - Sep 22, 2018	Live Event
SANS Network Security 2018	Las Vegas, NV	Sep 23, 2018 - Sep 30, 2018	Live Event
SANS Brussels October 2018	Brussels, Belgium	Oct 08, 2018 - Oct 13, 2018	Live Event
SANS Northern VA Fall- Tysons 2018	Tysons, VA	Oct 13, 2018 - Oct 20, 2018	Live Event
SANS Denver 2018	Denver, CO	Oct 15, 2018 - Oct 20, 2018	Live Event
SANS October Singapore 2018	Singapore, Singapore	Oct 15, 2018 - Oct 27, 2018	Live Event
Mentor Session - SEC503	Ankara, Turkey	Oct 31, 2018 - Dec 19, 2018	Mentor
Mentor Session - SEC503	Ballston, VA	Nov 01, 2018 - Dec 06, 2018	Mentor
SANS Dallas Fall 2018	Dallas, TX	Nov 05, 2018 - Nov 10, 2018	Live Event
SANS San Diego Fall 2018	San Diego, CA	Nov 12, 2018 - Nov 17, 2018	Live Event
San Diego Fall 2018 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Nov 12, 2018 - Nov 17, 2018	vLive
SANS Stockholm 2018	Stockholm, Sweden	Nov 26, 2018 - Dec 01, 2018	Live Event
Tactical Detection & Data Analytics Summit & Training 2018	Scottsdale, AZ	Dec 04, 2018 - Dec 11, 2018	Live Event
SANS Cyber Defense Initiative 2018	Washington, DC	Dec 11, 2018 - Dec 18, 2018	Live Event
SANS Security East 2019	New Orleans, LA	Feb 02, 2019 - Feb 09, 2019	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced