Global Information Assurance Certification Paper

Montgomery Toren

Intrusion Detection In Depth

San Diego, CA. October 2001

GIAC Practical Version: 3.0

11 February 2002

Assignment 1 - Describe the State of Intrusion Detection

This paper will describe the steps I went through to solve a problem I was having with false positives that were being caused by data in the referrer portion of the HTTP request. I never did find a satisfactory solution under Snort 1.7, but Snort 1.8 included new functionality that solved my problem.

Background:

Soon after installing Snort on my employer's network, I began to see a large number of false positives alerting on unsafe Front Page extensions and certain CGI scripts. Looking at the entire packet from the alerts (see fig. 1), it became clear that the unsafe extensions and scripts were not part of the requested URL to our site, but were actually in use by our partners. That is, the alerts were matching on the 'Referer' (misspelled as in the HTTP spec) portion of the HTTP request, which details the URL that was clicked to get to our site. It is a large e-commerce site with many hundreds of affiliates, and the mechanism that these affiliates used to refer traffic to our site varied widely from site to site. As it turns out, many of these sites were using Front Page extension, or notoriously unsafe CGI scripts such as phf or the email program mmstdod.cgi to refer traffic to our web site. They were actually scripts being used by our affiliates that were triggering Snort 1.7 alerts on our network.

Following is an example of how the mmstdod.cgi script would lead to false positives for our site.

For background, the mmstdod.cgi script is part of the MailMan web mail package, and is vulnerable to remote command execution in versions prior to 3.0.27 (see http://www.eeye.com/html/Support/Retina/RTHs/CGI_Scripts/572.html).

Here is the Snort 1.7 rule that alerted on someone accessing (or scanning for) the script mmstdod.cgi.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"BETA - Attempt at mmstdod.cgi
    -   if installed, verify it is newer than 3.0.26"; content:"mmstdod.cgi"; nocase;)
```

To break this down, this rule starts with 'alert', which means that this will be logged as an alert. Next, the rule matches the protocol 'TCP', which is the protocol that HTTP uses. Next, the '$EXTERNAL_NET any -> $HOME_NET 80) is saying that if the traffic originates from any external host on any port (typically would be an ephemeral

port above 1024) and is destined to a server on our network at port 80 (typically an HTTP server), then continue to process this alert. Next is the message that will be displayed if all parts of the alert match – in this case it is a warning to make sure one is running a version greater than 3.0.26. The next part is where all the false positives would come – we are simply looking for the content "mmstdod.cgi" anywhere in the packet. As I will show in the next section, this was showing up not in the HTTP request (URI), but in the 'Referer' portion of the packet. The last part of the rule is the directive 'nocase', which tells Snort not to care about upper or lower case in the match.

Here is an example packet from the mmstdod.cgi false alert. I had to create this alert, as I no longer have any of my old 1.7 alerts anymore. I created this alert using internal web servers. On the server named sneakers, I created a script called mmstdod.cgi which contained a link back to the server for the url http://sneakers/blah. What you see in this packet is the request for blah (the red portion) and 'Referer' section (in blue), which is the fake mmstdod.cgi script I created. The source IP address of MY.NET.7.117 is my web browser, which is what is actually making the request.

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

[**] BETA - Attempt at mmstdod.cgi - if installed, verify it is newer than 3.0.2
6 [**]
01/31-16:32:22.539089 MY.NET.7.117:4827 -> MY.NET.120.80:80
TCP TTL:126 TOS:0x0 ID:14977 IpLen:20 DgmLen:411 DF
***AP*** Seq: 0x5EDA9090  Ack: 0xAF7E14DC  Win: 0x1E16  TcpLen: 20
47 45 54 20 2F 62 6C 61 68 20 48 54 54 50 2F 31  GET /blah HTTP/1
2E 31 0D 0A 41 63 63 65 70 74 3A 20 69 6D 61 67  .1..Accept: imag
65 2F 67 69 66 2C 20 69 6D 61 67 65 2F 78 2D 78  e/gif, image/x-x
62 69 74 6D 61 70 2C 20 69 6D 61 67 65 2F 6A 70  bitmap, image/jp
65 67 2C 20 69 6D 61 67 65 2F 70 6A 70 65 67 2C  eg, image/pjpeg,
20 61 70 70 6C 69 63 61 74 69 6F 6E 2F 76 6E 64   application/vnd
2E 6D 73 2D 70 6F 77 65 72 70 6F 69 6E 74 2C 20  .ms-powerpoint,
61 70 70 6C 69 63 61 74 69 6F 6E 2F 76 6E 64 2E  application/vnd.
6D 73 2D 65 78 63 65 6C 2C 20 61 70 70 6C 69 63  ms-excel, applic
61 74 69 6F 6E 2F 6D 73 77 6F 72 64 2C 20 2A 2F  ation/msword, */
2A 0D 0A 52 65 66 65 72 65 72 3A 20 68 74 74 70  *..Referer: http
3A 2F 2F 73 6E 65 61 6B 65 72 73 2F 63 67 69 2D  ://sneakers/cgi-
62 69 6E 2F 6D 6D 73 74 64 6F 64 2E 63 67 69 0D  bin/mmstdod.cgi.
0A 41 63 63 65 70 74 2D 4C 61 6E 67 75 61 67 65  .Accept-Language
3A 20 65 6E 2D 75 73 0D 0A 41 63 63 65 70 74 2D  : en-us..Accept-
45 6E 63 6F 64 69 6E 67 3A 20 67 7A 69 70 2C 20  Encoding: gzip,
64 65 66 6C 61 74 65 0D 0A 55 73 65 72 2D 41 67  deflate..User-Ag
65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 34 2E 30  ent: Mozilla/4.0
20 28 63 6F 6D 70 61 74 69 62 6C 65 3B 20 4D 53   (compatible; MS
49 45 20 36 2E 30 3B 20 57 69 6E 64 6F 77 73 20  IE 6.0; Windows
4E 54 20 34 2E 30 29 0D 0A 48 6F 73 74 3A 20 73  NT 4.0)..Host: s
6E 65 61 6B 65 72 73 0D 0A 43 6F 6E 6E 65 63 74  neakers..Connect
69 6F 6E 3A 20 4B 65 65 70 2D 41 6C 69 76 65 0D  ion: Keep-Alive.
0A 0D 0A                                         ...

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

Fig 1

Here is the log from the web server, which shows that mmstdod.cgi was in no way actually requested – just the file "blah." You can also see that "blah" was not served by the result code, which is 404. Again, the source IP address of MY.NET.7.117 is my web browser, which is what is actually making the request.

```
MY.NET.7.117 - - [31/Jan/2002:16:32:22 -0800] "GET /blah HTTP/1.1" 404
287
```

Lastly, if it is configured to do so, Apache will log referrers in a separate log file. Here is the log from that file:

```
http://sneakers/cgi-bin/mmstdod.cgi -> /blah
```

This shows that the URL http://sneakers/cgi-bin/mmstdod.cgi referred the user to the URL /blah on this server.

It is clear that the user actually requested the URL /blah from our site and in no way requested the mmstdod.cgi script. The above request does not even constitute a reconnaissance attempt, as no information about the script on our servers was ever returned.

While this may not seem like a large problem, for our site, this meant several hundred to as many as hundreds of thousands of false positives a day – depending on the advertising campaign at the time.

Obviously, the simplest way to fix this would be to just disable the rules that cause the false alerts due to our affiliate and partner campaigns. This has at least two drawbacks. First, I would have to stop monitoring for a wide range of Front Page extensions and CGI scripts. While this would work, it would also leave me blind to real attackers scanning for some of these well know vulnerabilities. Second, disabling a rule after the fact can often be messy; as I had arrived at work in the morning and seen tens of thousands of log entries from a major affiliate web site that was popping up our site to their customers. This was time consuming to remove these entries after the fact from my database.

Because disabling the entire rule was not satisfactory, I set out to find a way to re-write the rules that would cut down on these false positives. I came up with three basic techniques – all of which opened me up to complete circumvention (i.e. false negative) from a skilled attacker.

Technique #1 (For Snort 1.7)

The first idea I had was to use the regex expression in a Snort rule, coupled with a pass action to beat the false positive problem. I could create a new rule, above the existing rule, that matched on Referer*mmstdod.cgi, then did nothing (pass). That is, the rule would match a packet that had the mmstdod.cgi after the Referer keyword, then pass. Later in the ruleset would be the normal rule that matched the way it always did, but would only be reached if the pass rule was never activated. That is, there was a pass rule that would match on packets that had the script after the Referer, and then Snort would exit and not continue trying to match. If the pass rule did not match (evil script was in the request, not the Referer section), then the original alert rule would be reached and an alert would be issued. To use this, Snort had to be started with the –o command line option, which told it to change the order of rules to match pass rules first.

New Rule:

```
pass tcp $EXTERNAL_NET any -> $HOME_NET 80
(content:"Referer*mmstdod.cgi"; regex
; nocase;)
```

This says to match on a packet that contains the string Referer, followed by zero or more characters (the *), followed by mmstdod.cgi. If it finds a packet like this, it should pass it (and stop looking for other matches because of the command line -o). The "*" would match the referring web site.

This rule worked fine, but when I analyzed it more, I realized it also could be bypassed by a determined hacker. Since unknown arguments to a CGI script are typically ignored, an attacker who knew I had modified my rules in this way would only need to tack on a ?Referer-any text-mmstdod.cgi after the script in question. That is, if the attacker were scanning for the mmstdod.cgi script on my server with a request such as GET /cgi-bin/mmstdod.cgi, he would only have to change the URL to GET /cgi-bin/mmstdod.cgi?Referer-any text-mmstdod.cgi to fool my rule and have a false negative. If he were simply scanning to see if the script were on my site (Whisker style), he would get the same response from the server with or without the 'Referer' argument. If the attacker were actually attempting to exploit the script, there is a chance that adding the argument would ruin his exploit, but more often than not, CGI scripts will simply ignore arguments that aren't recognized.

Technique #2 (For Snort 1.7)

To improve on the regex idea, I found a tip in the Snort user's guide to limit how deep in the packet to look for the payload. I could then create a rule that would be effective and very difficult to bypass. Limiting the depth to 20, for example, should keep the search from going into the referrer section of the packet.

Before:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"BETA - Attempt at
mmstdod.cgi
      -if installed, verify it is newer than 3.0.26";
content:"mmstdod.cgi"; nocase;)
```

After:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"BETA - Attempt at
mmstdod.cgi
      -if installed, verify it is newer than 3.0.26";
content:"mmstdod.cgi"; depth:20
 ;nocase;)
```

Notice the addition of the 'depth:20' directive – this is telling Snort to not look past the first 20 characters.

However, the problem is that this could, in theory, be circumvented by a hacker who knew you were doing this, thus leading to a false negative. How would a hacker do this? There are multiple ways to specify the same URL – all of which will end up being valid to the web server. One way is simply to add many '/////'s to the URL – which many web servers (including Apache) will ignore. The problem is, the web server ignores them, but they push the evil script name in question out past the 20 byte mark. To show an example of this, I have set up tcpdump to collect traffic when I make the web request

http://MY.NET.2.158//
////////////////////////////////mmstdod.cgi

```
17:41:22.412091 MY.NET.7.117.4965 > MY.NET.2.158.80: P 0:462(462) ack 1 win 8760
 (DF)
0x0000   4500 01f6 d93c 4000 7d06 c081 ac16 0775        E....<@.}......u
0x0010   ac1a 029e 1365 0050 9d66 f813 5fa7 72b7        .....e.P.f.._.r.
0x0020   5018 2238 b7da 0000 4745 5420 2f2f 2f2f        P."8....GET.////
0x0030   2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f        ////////////////
0x0040   2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f        ////////////////
0x0050   2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f        ////////////////
0x0060   2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f        ////////////////
0x0070   2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f        ////////////////
0x0080   2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f        ////////////////
0x0090   2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f        ////////////////
0x00a0   2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f2f 2f6d        ///////////////m
0x00b0   6d73 7464 6f64 2e63 6769 2048 5454 502f        mstdod.cgi.HTTP/
0x00c0   312e 310d 0a41 6363 6570 743a 2069 6d61        1.1..Accept:.ima
0x00d0   6765 2f67 6966 2c20 696d 6167 652f 782d        ge/gif,.image/x-
0x00e0   7862 6974 6d61 702c 2069 6d61 6765 2f6a        xbitmap,.image/j
0x00f0   7065 672c 2069 6d61 6765 2f70 6a70 6567        peg,.image/pjpeg
0x0100   2c20 6170 706c 6963 6174 696f 6e2f 766e        ,.application/vn
0x0110   642e 6d73 2d70 6f77 6572 706f 696e 742c        d.ms-powerpoint,
0x0120   2061 7070 6c69 6361 7469 6f6e 2f76 6e64        .application/vnd
0x0130   2e6d 732d 6578 6365 6c2c 2061 7070 6c69        .ms-excel,.appli
0x0140   6361 7469 6f6e 2f6d 7377 6f72 642c 202a        cation/msword,.*
0x0150   2f2a 0d0a 4163 6365 7074 2d4c 616e 6775        /*..Accept-Langu
0x0160   6167 653a 2065 6e2d 7573 0d0a 4163 6365        age:.en-us..Acce
0x0170   7074 2d45 6e63 6f64 696e 673a 2067 7a69        pt-Encoding:.gzi
0x0180   702c 2064 6566 6c61 7465 0d0a 5573 6572        p,.deflate..User
0x0190   2d41 6765 6e74 3a20 4d6f 7a69 6c6c 612f        -Agent:.Mozilla/
0x01a0   342e 3020 2863 6f6d 7061 7469 626c 653b        4.0.(compatible;
0x01b0   204d 5349 4520 362e 303b 2057 696e 646f        .MSIE.6.0;.Windo
0x01c0   7773 204e 5420 342e 3029 0d0a 486f 7374        ws.NT.4.0)..Host
0x01d0   3a20 696e 7472 7564 6572 0d0a 436f 6e6e        :.intruder..Conn
0x01e0   6563 7469 6f6e 3a20 4b65 6570 2d41 6c69        ection:.Keep-Ali
0x01f0   7665 0d0a 0d0a                                 ve....
```

This is a completely valid URL to Apache, and it will serve the file if it exists. As can be seen, the match string of 'mmstdod.cgi' is now deep into the packet, and is as deep as the referrer section would be in a normal packet, so it would be impossible to adjust the depth to not false positive, and at the same time, not be vulnerable to a false negative.

If one turns on the http_decode preprocessor (which most people do), it will attempt to 'normalize' the URL, which is to remove the alternate representations such as this. In this particular example, the http_decode preprocessor did catch the extra '///'s and removed them, so this alert would not get past Snort. However, with all the combinations of Unicode character equivalents and tricks such as ../../../ that could be introduced (depending on the web server), I worried that someone would be able to sneak some

padding past Snort. This may seem overly paranoid, but this was at the time when programs like Whisker were getting past many IDS's. For Whisker, see http://www.wiretrip.net/rfp/p/doc.asp/i7/d21.htm

Technique #3 (For Snort 1.8)

My first idea for reducing the false positives in Snort 1.8 was to use the new "content-list" directive. This allowed matching on multiple patterns in a packet, so I could write a rule that matched on "mmstdod.cgi" but also had to avoid a match on the pattern !"Referer". In other words, the packet would have to contain mmstdod.cgi (the script in question), but couldn't contain "Referer" in it – thus solving my false positive problems.

Before:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"BETA - Attempt at
mmstdod.cgi
   - if installed, verify it is newer than 3.0.26";
     content:"mmstdod.cgi"; nocase;)
```

After:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"BETA - Attempt at
mmstdod.cgi
   - if installed, verify it is newer than 3.0.26"; content-
     list:"mmstdod.cgi" !"Referer"; nocase;)
```

Again, this method was open to false negatives, as the URL could be crafted to avoid detection. The way around this rule is identical to technique #2 from Snort 1.7 – simply add a fake argument of ?Referer to the end of the URL.

Technique #4 (for Snort 1.8)

The answer seems to have been created just for this use – the new uricontent directive. This directive instructs Snort to only look at the URI portion of the packet (the GET followed by the URL that was requested). For the definition of URI, see http://www.ietf.org/rfc/rfc2396.txt

This means that Snort would stop looking after the GET, and would never reach the Referer section to be confused by. This was perfect. The manual for Snort 1.8 really undersold this new feature. The manual's notes state, "The uricontent rule allows searches to be matched against only the URI portion of a request. This allows rules to search only the request portion of an attack without false alerts from server data files." See (http://www.snort.org/docs/writing_rules/chap2.html#tth_sEc2.3.30)

Typically, you can avoid being alerted from server data files by carefully defining the home net to include your web servers, then making sure that the rules are directional to only match traffic TO your home net. The real use of the uricontent directive, in my

opinion, is for exactly what my problem had been – to ignore other innocent non-URI related traffic in the HTTP request.

It is clear that uricontent is the directive of choice for web based rules in Snort 1.8. In fact, all the old web rules that match on URI content have been re-written in Snort 1.8 to use the new uricontent rather then 'content'. Here is the new Snort 1.8 rule for mmstdod.cgi (as supplied by the standard Snort distribution).

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI mmst
dod.cgi access"; uricontent:"/mmstdod.cgi"; nocase;
flags:a+;classtype:attempted
-recon; sid:819; rev:1;)
```

This is essentially the same as the old 1.7 rule, except it uses the 'uricontent' keyword rather than 'content' to match on the mmstdod.cgi text.

It should be noted that the uricontent directive will not work if the http_decode preprocessor is not enabled. This preprocessor is required for Snort to break up the request and keep track of where the URI ends. For reference to this, see http://archives.neohapsis.com/archives/snort/2001-11/0177.html

Assignment 2- Network Detects

Detect 1 – Attempt to misuse 'formmail.pl' CGI script (possibly to spoof email SPAM).

Source of Trace

This event was recorded on the 'DMZ' directed at one of our public web servers.

Detect was generated by:

The device that triggered the alert was Snort version 1.8.1.

The rule that generated the alert is as follows:

web-cgi.rules:alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI form mail access";flags: A+; uricontent:"/formmail"; nocase; reference:cve,CVE-1999-0172; reference:arachnids,226; classtype:attempted-recon; sid:884; rev:1;)

This rule is simply looking for the content "/formmail" anywhere in an HTTP request (URI). This script would typically be in a CGI type directory.

The alert generated by Snort (and displayed by ACID) is as follows:

```
--------------------------------------------------------------------------------
#(1 - 101642) [2001-11-15 22:09:54] [CVE/CVE-1999-0172] [arachNIDS/226]  WEB-CGI formmail
access
IPv4: 63.234.21.210 -> MY.NET.121.32
      hlen=5 TOS=0 dlen=368 ID=31889 flags=0 offset=0 TTL=112 chksum=46105
TCP:  port=2237 -> dport: 80  flags=***AP*** seq=2163387760
      ack=1472891879 off=5 res=0 win=8760 urp=0 chksum=28721
Payload:  length = 328

000 : 47 45 54 20 2F 63 67 69 2D 62 69 6E 2F 66 6F 72   GET /cgi-bin/for
010 : 6D 6D 61 69 6C 2E 70 6C 3F 72 65 63 69 70 69 65   mmail.pl?recipie
020 : 6E 74 3D 72 6D 69 74 63 68 65 6C 6C 39 36 30 31   nt=rmitchell9601
030 : 40 61 6F 6C 2E 63 6F 6D 26 73 75 62 6A 65 63 74   @aol.com&subject
040 : 3D 68 74 74 70 3A 2F 2F 77 77 77 2E FF FF FF FF   =http://www.ours
050 : FF FF FF 2E 63 6F 6D 2F 63 67 69 2D 62 69 6E 2F   ite.com/cgi-bin/
060 : 66 6F 72 6D 6D 61 69 6C 2E 70 6C 26 65 6D 61 69   formmail.pl&emai
070 : 6C 3D 65 6C 69 74 65 5F 68 66 6F 34 72 79 37 40   l=elite_hfo4ry7@
080 : 6D 73 6E 2E 63 6F 6D 26 3D 68 74 74 70 3A 2F 2F   msn.com&=http://
090 : 77 77 77 2E FF FF FF FF FF FF FF 2E 63 6F 6D 2F   www.oursite.com/
0a0 : 63 67 69 2D 62 69 6E 2F 66 6F 72 6D 6D 61 69 6C   cgi-bin/formmail
0b0 : 2E 70 6C 20 48 54 54 50 2F 31 2E 31 0D 0A 41 63   .pl HTTP/1.1..Ac
0c0 : 63 65 70 74 3A 20 69 6D 61 67 65 2F 67 69 66 2C   cept: image/gif,
0d0 : 20 69 6D 61 67 65 2F 78 2D 78 62 69 74 6D 61 70    image/x-xbitmap
0e0 : 2C 20 69 6D 61 67 65 2F 6A 70 65 67 2C 20 69 6D   , image/jpeg, im
0f0 : 61 67 65 2F 70 6A 70 65 67 2C 20 2A 2F 2A 0D 0A   age/pjpeg, */*..
100 : 55 73 65 72 2D 41 67 65 6E 74 3A 20 4D 69 63 72   User-Agent: Micr
110 : 6F 73 6F 66 74 20 55 52 4C 20 43 6F 6E 74 72 6F   osoft URL Contro
120 : 6C 20 2D 20 36 2E 30 30 2E 38 38 36 32 0D 0A 48   l - 6.00.8862..H
130 : 6F 73 74 3A 20 77 77 77 2E 6E 65 74 66 6C 69 78   ost: www.oursite
140 : 2E 63 6F 6D 0D 0A 0D 0A                           .com....
```

The interesting fields are as follows:

The source IP address is 63.234.21.210 (0-1pool21-210.nas1.shreveport1.la.us.da.qwest.net), owned by Qwest Communications (NETBLK-NET-QWEST-63BLKS)

950 17th St. Suite 1900
Denver, CO 80202
US

The destination IP address is my server.

The ID of 31889 doesn't stand out as strange, and a quick search of Google for this ID didn't turn anything up. That is, this ID does not appear to be associated with a well know packet crafting tool.

The TTL of 112 is normal, and it probably started with an initial TTL of 128 (16 hops is typical), which would likely make this a Windows box.

The source port of 2237 is a typical ephemeral port – especially for a Windows application.

The destination of port 80 is the HTTP port on my web server.

The flags ACK and PSH are set. This means it is an established TCP session (ACK), and the PSH simply means that the application wants the data to be processed immediately and not to wait for buffers to fill.

This all adds up to a normal TCP connection. The really interesting part is the content of the payload -- GET /cgi-bin/formmail.pl?recipient=rmitchell9601@aol.com&subject=http://www.oursite.com/cgi-bin/formmail.pl&email=elite_hfo4ry7@msn.com&=http://www.oursite.com/cgi-bin/formmail.pl HTTP/1.1

As this shows, the attacker appears to be using a well-known vulnerability in formmail.pl to send himself an email from our server. If the email arrives, the attacker knows that we are running an unpatched formmail.pl on our server. This is one of my favorite exploits to see, as the attacker actually gives away his email address! To date, I have collected 76 unique email addresses of people attempting this exploit.

Probability the source address was spoofed:

As the reply to this attempt is sent via email from the web server itself, this is one of the rare attacks (other than a denial of service) where the source IP address could be spoofed. But still, completing a three-way TCP handshake (which is required by the web server before it will accept the GET request) is extremely difficult to do with a forged source IP address. There are several possibilities for this. One possibility is that the attacker could have control of a machine close to the IP address he is spoofing (to sniff sequence numbers off the wire as they pass by) and then use these sequence numbers in forged packets. However, this is a lot of work to hide an attack that already requires giving a valid email address. Considering that the attacker didn't even bother to see if the script existed on my server before attempting to exploit it (which it didn't), or even if I had a cgi-bin directory on the web server (again, I didn't), I get the impression that he was not skilled enough to pull off a spoofed three-way handshake. If this alert had just happened, I would traceroute back to the attacking host from my web server, and see if the TTL in the packet header (in this case 112) plus the number of hops, added up to a common starting TTL. It appears the starting TTL was 128 (probably Windows), and it took 16 hops to get to my web server, but because far too much time has passed (2 months), this test would be inconclusive. Again, spoofing is possible in this instance, as the response comes via email, but it is still unlikely that the source was spoofed. However, I would need more information to be completely sure.

Attack Mechanism:

The attack works by completing the TCP three-way handshake, then sending an HTTP GET to the server. In this case, the request looks like this: GET /cgi-bin/formmail.pl?recipient=rmitchell9601@aol.com&subject=http://www.oursite.com/cgi-bin/formmail.pl&email=elite_hfo4ry7@msn.com&=http://www.oursite.com/cgi-bin/formmail.pl HTTP/1.1

The GET appears to set the recipient value to rmitchell9601@aol.com, which is not nearly as likely to be a faked as a Hotmail or a Yahoo mail account. It also sets a fake sender address (elite_hfo4ry7@msn.com) and the subject is set to the URL (our server) that is being tested. The subject is probably chosen so that the attacker can keep track of which servers he tested when the email comes back showing the server vulnerable.

According to this entry in the Security Focus vulnerability database http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=2469, this vulnerability can be used to send anonymous SPAM, as no indication of the original sender (via the CGI interface) will be in the email. That means the attacker could possibly send SPAM that appeared to come from our web servers.

Correlations:

This is a well-known vulnerability, which is given the Bugtraq ID 2469 **:** **http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=2469**. This is also CVE-1999-0172 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0172

Also, this was discussed on the Incidents mailing list, where someone saw the exact same signature that I have been seeing from multiple sources. http://archives.neohapsis.com/archives/incidents/2001-08/0446.html The fact that someone has seen the exact same request (including the strange User-Agent "Microsoft URL Control - 6.00.8862" which he thinks is a VB script) leads me to believe that there is a script available to SPAMers that will automatically scan for vulnerable hosts to exploit. Also, I have seen this exact same signature 76 times recently. I did not find the script that is causing my signature, although I did find several scripts out there that exploit formmail – including a much more severe one that uses a different vulnerability in it to open an XTERM back to the attacker.

Evidence of active targeting:

This was very definitely actively targeted. First, our site does not use formmail, so it could not have been from a broken page on our own server. Second, the recipient and subject of the message are hard-coded to alert the attacker if the test was successful.

Severity:

We calculate Severity using:
(Criticality + Lethality) - (System countermeasures + Network countermeasures)

Each item is marked from 1 – 5, 5 being the highest, 1 being the lowest.

Criticality = 3

This was a publicly available server in a protective DMZ.

Lethality = 3
While embarrassing to send out SPAM, this exploit would not have lead to system or network penetration, or any exposure of critical information. I would have given this a lower criticality of 2, except that our company sends a large amount of email to our customers, and therefore we can't afford to be placed on any open relay or spammer blackhole list.

System countermeasures = 5
This was a fully patched web server that had never had frommail.pl installed.

Network countermeasures = 4
The network was allowing access to port 80 only, and the web server had no outbound access to the Internet. Even if the script had been on the server, the email would never have gotten outbound past the firewall.

(3+3) – (5+4) = -3

Defensive Recommendations:

Periodically use scanning tools (such as Webtrends, Whisker, etc) to scan for vulnerable CGI scripts on web servers. In this case we were fine, as the script had never been installed on our servers

Multiple choice test question:

Someone scanning for the formmail.pl script on your web server is likely trying to do what?

A) Send email to your website.
B) Exploit the Unicode bug
C) Hide the real source of their SPAM by making it look like it came from your site.
D) None of the above

Answer: C.

Detect 2: MISC loopback traffic (repeated connects from my mail servers to 127.0.0.100)

Source of Trace

This event was recorded on the internal network. It was coming from a mail server that sends newsletters and shipping alerts to our customers on the Internet. This mail server is not accessible from the Internet – it only has outbound access through the firewall.

Detect was generated by:

This was detected by Snort 1.8.1.

The rule that generated the alert is as follows:

alert ip any any <> 127.0.0.0/8 any (msg:"MISC loopback traffic"; classtype:bad-unknown; sid:528; rev:1;)

This rule will alert on any traffic to or from the reserved network 127.0.0.0/8 (the loopback network). This would always be suspicious, as packets should never be routed to this network.

The actual Snort alert (as displayed by ACID) is a follows:

```
---------------------------------------------------------------------------
#(2 - 26553) [2002-01-24 09:59:46]  MISC loopback traffic
IPv4: MY.NET.2.107 -> 127.0.0.100
    hlen=5 TOS=0 dlen=44 ID=59411 flags=0 offset=0 TTL=255 chksum=26062
TCP:  port=42242 -> dport: 25  flags=******S* seq=1221679726
    ack=0 off=6 res=0 win=8760 urp=0 chksum=65449
        Options:
    #1 - MSS len=4 data=05B4
Payload: none
```

The interesting fields are as follows:

The source IP address is MY.NET.2.107 (my internal mail server).

The destination IP address is 127.0.0.100 (on the loopback reserved network).

The ID of 59411 doesn't stand out as strange, and a quick search of Google for this ID didn't turn anything up. That is, this ID does not appear to be associated with a well-known packet crafting tool.

The TTL of 255 is normal for a Solaris server, which this mail server is. This confirms that the alert is being detected on the source network, as the TTL has not been decremented from its maximum possible value of 255.

The source port of 42242 is a typical ephemeral port for Solaris – especially for the sendmail application.

The destination port is 25, the normal SMPT port. Since sendmail was running on this server, my initial guess was that the application sending these packets was probably sendmail.

The only flag set is SYN, which means this is an initial request for a connection (first phase of the three-way handshake). There are no other alerts with anything except SYN, so the connections were not succeeding.

Other Logs:

I also have a sendmail log from the mail server. Several things pointed to the fact that sendmail created this traffic, including the fact that the destination port was 25 (the SMTP port) and that sendmail is the primary application running on this server. Also, a source port of 42242 is a common source port for sendmail to use.

```
Jan 22 09:59:43 mx7 sendmail[19875]: JAA19304: to=<berry@southeast.net>, delay=3
+00:19:33, xdelay=00:03:44, mailer=esmtp, relay=null.southeast.net. [127.0.0.100
], stat=Deferred: Connection timed out with null.southeast.net.
```

Lastly, I have the MX record associated with southeast.net (the MX hosts to which sendmail is attempting to send mail).

```
myserver% dig -q-type mx southeast.net

; <<>> DiG 8.3 <<>> -q-type mx southeast.net
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; QUERY SECTION:
;;      southeast.net, type = MX, class = IN

;; ANSWER SECTION:
southeast.net.          5h57m7s IN MX   10 null.southeast.net.

;; ADDITIONAL SECTION:
null.southeast.net.     5h57m7s IN A    127.0.0.100

;; Total query time: 3 msec
;; FROM: myserver to SERVER: default -- MY.NET.120.109
;; WHEN: Thu Jan 24 17:40:20 2002
;; MSG SIZE  sent: 31  rcvd: 68
```

This shows that the only MX host for the domain (null.southeast.net) has an IP address of 127.0.0.100! This is clearly an illegal IP address.

Probability the source address was spoofed:

Almost none. The source address is my own sendmail server, from which I have the collaborating sendmail log file.

Attack Mechanism:

This isn't really an attack – it is clearly a misconfiguration by the DNS admin of southeast.net. The reason I'm analyzing this is because this technique could be used as an attack. When I first saw this alert from Snort three months ago, the site that was misconfigured then was medianone.net, which was setting its MX server to 127.0.0.0. I saw the alert, and wondered if someone had installed a Trojan or a back door on my mail server, and was attempting to use it for a Denial of Service attack, or to scan my internal network, or possibly to open a covert channel. The real giveaway was that the connections were all to port 25, which is the SMTP port. Also, this Snort rule was set to alert on any of the TCP flags (not just established connections), so the fact that it was only logging the initial SYN attempts showed that the connections were failing.

What it turned out to be, which is the same thing as is happening in this alert, is that the DNS administrator for the domain had set the MX record for the domain to be a loopback address. This may be because he didn't want any more email, or more likely, he had made an honest mistake. But the fact that my sendmail server had simply trusted DNS and had connected to an illegal IP address got me thinking. What would stop someone from setting the MX record to the IP address of one of my internal networks? Could this be used possibly to connect to an internal mail server that isn't meant to be connected to from the Internet?

I don't think there is anything that would stop a person from doing this. With some reconnaissance or inside information, an attacker could know about an internal mail server that either hasn't been properly patched (since the admin might feel it is safe from the Internet), or contains sensitive mailing lists that shouldn't be sent to except from inside the company. Another possibility, although more remote, would be an internal server with a custom application that is running on port 25. At any rate, this vulnerability would allow an attacker with control of a domain's DNS information to make connections to port 25 of internal networks.

Getting the internal server to initiate the email would not be very difficult – I can think of two possible ways. First, an attacker may be able to send an email to the company with a bogus recipient. If the server that makes the decision to bounce the message back to the sender is a machine with internal network access, the bounced message would generate a network connection to the IP address of the MX record over which the attacker has control. The second method would be the reason that we keep getting these alerts – we allow people to sign up for a newsletter. When the newsletter is sent, it will go to whatever email address was specified when they signed up. Probably from a combination of typos and just the very large number of domains to which we attempt to connect, I actually see this exact problem (connecting to a loopback address) fairly often.

Correlations:

From my own logs, there is a correlation between Snort and my sendmail logs. Also, the alert comes up every 15 minutes in Snort, as the message keeps trying to be redelivered by sendmail until it expires after 5 days (the default queue retry in sendmail is every 15 minutes).

As far as correlations on the Internet, I found nothing about it by searching the Incidents mailing list, or from a casual search on Google. This isn't to say that I think I am the first to think of this attack, or to see this problem, but I'm not seeing info on it in any of the regular security sources.

Evidence of active targeting:

In this case, there is very little evidence of active targeting. Unless the admins at southeast.net were setting out to annoy me with Snort alerts, I feel this was most likely a simple mistake. But then again, I can't be positive of that. For reference, here is the owner of the domain:

Registrant:
Southeast Network Services (SOUTHEAST2-DOM)
390 N Orange Ave, Suite 2000
Orlando, FL 32801
US

Severity

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

This will be calculated on the worst-case basis (i.e. if that had been a targeted attack to reach internal servers on my network as I described in the Attack Mechanism section).

Criticality = 4
These are important mail servers.

Lethality = 2
The possible attacks I came up with are all difficult to do, and fairly low impact even if they work.

System Countermeasures = 4
Sendmail is patched on all servers, and Bind is up to date. There is not much more I can do, short of changing Bind not to accept IP addresses that it shouldn't for MX records.

Network Countermeasures = 4
The mail server was not accessible from the Internet – not even to port 25 (SMTP). This is an outbound email server only, which is on a DMZ that does not allow it to connect back to internal hosts.

(4+2) – (4+4) = -2

Defensive Recommendations:

The real answer would be for the DNS server (typically Bind) to be intelligent enough to not accept internal IP addresses or loopback addresses from external domains. If this isn't feasible, then a good answer would be to place email servers that could possibly be reached by either of the two methods I described (bounces or opt-in mailings) in a protected DMZ where they can't reach other internal servers.

Multiple choice test question:

Your NIDS system is alerting on your email server attempting connections to the loopback network (but not its own loopback address). What is likely happening?

A) Your mail server is trying to send email to itself.
B) Your mail server has been compromised, and is being used in a denial of service attack.
C) The MX record for a domain you are sending mail to has been set to an illegal loopback address.
D) There is a covert channel on which your mail server is communicating.

Answer: C.

Detect 3: MISC data in TCP SYN packet (3DNS)

Source of trace:

This event was recorded on the 'DMZ' directed at my external (public) DNS server.

Detect was generated by:

The device that generated the alert was Snort 1.8.1.

The Snort rule that generated this alert is as follows:

```
misc.rules:alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC data in TCP S
YN packet"; flags:S; dsize:>6;  sid:526; rev:1;)
```

This rule is looking for any packet with a lone SYN flag set (flags:S) with a payload size greater than 6 (dsize:>6). This is a strange packet, as a lone SYN usually means the start of a three-way handshake (which would be followed by a SYN-ACK by the receiving host, then an ACK from the sending host) and initial SYN packets do not normally carry any data. But as you can see from this Snort alert, this packet contained 24 bytes of zeros.

```
---------------------------------------------------------------------------
#(1 - 1389) [2001-07-30 03:06:21]  MISC data in TCP SYN packet
IPv4: 212.62.14.150 -> MY.NET.2.154
    hlen=5 TOS=0 dlen=64 ID=3 flags=0 offset=0 TTL=53 chksum=62508
TCP:  port=22809 -> dport: 53  flags=******S* seq=4125342834
```

```
    ack=2030984515 off=5 res=0 win=2048 urp=0 chksum=13387
Payload:  length = 24

000 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.................
010 : 00 00 00 00 00 00 00 00                    ........
```

The interesting fields are as follows:

The source IP address is 212.62.14.150 (3dns-1.itronics.ie), owned by Itronics of Great Britain.

The destination IP address is my public DNS server.

The ID of 3 is suspicious. The chance of randomly seeing such a low ID number is very small.

The TTL of 52 is normal, and it probably started with an initial TTL of 64 (12 hops is typical), which would likely make this a Linux box.

The source port of 22809 is a typical ephemeral port – especially for a Linux application.

The window size of 2048 is a bit small, but expected since this is a transatlantic packet.

The destination of port 53 is the DNS (bind) port on my web server.

The flags are sent to SYN only. This is very strange, as there is a payload (24 bytes of zeros). Normally, a SYN is sent at the beginning of a TCP three-way handshake, and no data would be present.

This packet shows a connection to TCP port 53, which would normally be for a DNS zone transfer.

This packet really stumped me when I first saw it. In retrospect, I suppose a host name of 3dns-1.itronics.ie should have really tipped me off to what was going on. As it turns out, this is a probe sent from F5's 3DNS product to determine how quickly my DNS server responds.

Probability the source address was spoofed:

Almost none. The point of this probe is to calculate round trip times to our server; so getting a reply (the SYN-ACK that we would send) would be required.

Attack Mechanism:

The remote 3DNS server would periodically send these probes to our DNS server. According to Dan Hawrylkiw (see Incidents posting at

http://www.securityfocus.com/archive/75/250180), this traffic is used by global load balancers to determine which of their cache sites are closest to their clients. They probe the DNS server because they know that a DNS server will often be open to the Internet on at least port 53. When the probe sender gets DNS requests from a site they have previously probed, the product will issue the IP address of the mirror or cache site that got the fastest response from their earlier probes.

For a period of time I was seeing many of these requests, but they have now stopped. Maybe they realized that the addition of data on the SYN was tripping IDS systems.

I have also seen a similar technique from Akamai, where they make a connection to my DNS server with the IP option TS set (timestamp). They are apparently trying to accomplish the same thing with a timestamp.

Correlations:

A paper titled ***Interpreting Network Traffic*** by Richard Bejtlich (http://home.satx.rr.com/bejtlich/intv2-8.html) goes into detail about this technique.

Dan Hawrylkiw http://www.securityfocus.com/archive/75/250180

Evidence of active targeting:

This is not really an attack, but it certainly was actively targeted to our DNS server. I did not see any similar probes to any of our other DMZ machines, so they probably found our DNS server from Internic records.

Severity:

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

Criticality = 4
Our external DNS server is critical.

Lethality = 0
This is not meant as an attack. Since the number of probes was never excessive, this was essentially harmless.

System Countermeasures = 5
This DNS server has all recommended OS and Bind patches applied, and is configured not to allow zone transfers from non-authorized hosts.

Network Countermeasures = 4

There is a firewall that prevents access to any port other than 53. It does allow TCP connections to port 53 from anywhere, which could be limited to just the secondary DNS servers, but I am using the Bind configuration to enforce zone transfer restrictions.

(4+0) – (5+4) = -5

Defensive Recommendations:

If the probes were a bother, one could block TCP access to port 53 on the DNS server except from our know Secondary (slave) servers. In general, TCP is only used for zone transfers with DNS, and you should know which servers are allowed to do this.

Also, if the probes are coming from a small number of predictable hosts, these can be blocked in the firewall.

Multiple choice test question:

Seeing periodic data on SYN directed at your DNS service is likely caused by the following:

A) A global load balancer
B) A broken router
C) A covert channel hiding data in the SYN packet
D) None of the above

Answer: A.

Detect 4: SYN-FIN scan of port 22 (to find SSH servers).

Source of trace

This event was recorded outside the firewall (on the same VLAN as the external interface of the firewall).

Detect was generated by:

The device that generated the alert was Snort 1.8.1, with ACID to display the data.

The alert was not generated by a Snort signature, but rather by the stream4 preprocessor within Snort. This preprocessor is meant to provide stateful inspection/stream reassembly for Snort. This is the preprocessor, for example, that would take all the individual packets from an interactive TELNET session, and put them back together so an alert could be made from the normal content rules. This is also the preprocessor that is

used to catch stealth activity – which is why it caught this SYN FIN scan. Here is a summary of the scan activity.

```
spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection 2001-11-06 19:13:05 212.28.154.100:22  MY.NET.131.249:22  TCP
spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection 2001-11-06 19:13:05 212.28.154.100:22  MY.NET.131.237:22  TCP
spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection 2001-11-06 19:13:04 212.28.154.100:22  MY.NET.131.218:22  TCP
spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection 2001-11-06 19:13:04 212.28.154.100:22  MY.NET.131.217:22  TCP
spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection 2001-11-06 19:13:04 212.28.154.100:22  MY.NET.131.204:22  TCP
spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection 2001-11-06 19:13:04 212.28.154.100:22  MY.NET.131.192:22  TCP
spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection 2001-11-06 19:13:03 212.28.154.100:22  MY.NET.131.173:22  TCP
spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection 2001-11-06 19:13:03 212.28.154.100:22  MY.NET.131.167:22  TCP
```

Here are a few of the alerts in greater detail (the first 2):

```
--------------------------------------------------------------------------------
#(1 - 93545) [2001-11-06 19:13:05]  spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection
IPv4: 212.28.154.100 -> MY.NET.131.249
    hlen=5 TOS=0 dlen=40 ID=39426 flags=0 offset=0 TTL=18 chksum=17456
TCP:  port=22 -> dport: 22  flags=******SF seq=347235885
    ack=882285820 off=5 res=0 win=1028 urp=0 chksum=38049
Payload: none

--------------------------------------------------------------------------------
#(1 - 93543) [2001-11-06 19:13:05]  spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection
IPv4: 212.28.154.100 -> MY.NET.131.237
    hlen=5 TOS=0 dlen=40 ID=39426 flags=0 offset=0 TTL=18 chksum=17468
TCP:  port=22 -> dport: 22  flags=******SF seq=347235885
    ack=882285820 off=5 res=0 win=1028 urp=0 chksum=38061
Payload: none

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

These packets appear crafted in numerous ways.

First, they all share the same ID (39426). Also, the source port (22) is the same as the destination port (22), which is not normal. Typically, the source port should be greater than 1024.

The TTL of 18 seems a bit strange as well. This probably means it started at 32, which is lower than most TCP stacks default to.

The window size of 1028 is also smaller that is typically be seen.

The flags are also illegal, as both SYN and FIN are set at the same time. Normally, a SYN is seen when a TCP connection is starting, and a FIN is seen when the connection is terminating. They should never be set in the same packet.

This particular ID number (39426) from all of the alerts is associated with a tool named synscan. Here is a description from Daniel Martin from his posting on Incidents http://www.securityfocus.com/archive/75/221281

> "Telltale signs of a synscan variant:

- SYN+FIN scan followed by an almost immediate regular SYN to open hosts. (Doesn't this rather defeat the point of running a SYN+FIN scan in the first place? Isn't it the point of a SYN+FIN scan to avoid being detected by those hosts that aren't running a firewall but do log regular connections to open ports?)
- IP id 39426 (This is what you get when your source code says
  ip->id = 666;
  and you compile on a little-endian machine, like intel-based linux boxes) on the SYN+FIN scan
- Source port == Destination port (again, on the SYN+FIN scan - the synscan program uses this to distinguish FIN responses from open scanned machines from other unrelated incoming FIN packets)"

The only difference between this description and what I was seeing is that there was no secondary connection to open ports. This, however, is explained by the fact that my firewall is not allowing SSH from the Internet, so the scanner would not see any open ports to try the secondary connection to.

Probability the source address was spoofed:

This is probably not spoofed, as the attacker is looking to get reconnaissance data back from the scan. It would be extremely difficult to spoof the source address and still manage to get the results. However, one possible method would be for the attacker to have control of a machine close to the IP address he is spoofing (to sniff sequence numbers off the wire as they pass by) and then use these sequence numbers in forged packets.

Attack Mechanism:

The attacker is apparently using the synscan tool to scan a range of IP addresses (all of my IP address were scanned). The tool uses a SYN FIN scan to be stealthy (i.e. many systems will not log a connection of this type), and also because some packet filtering firewalls (non stateful) will actually let these packets through even if they are configured to block traffic to the port in question. If Daniel Martin's description of the tool is accurate, it would then do a regular SYN to the open hosts, and log the successes to a file. This file would then presumably be used by some other tool (like the RAMON worm did with other vulnerable services) to attempt an exploit of the SSH service. There have been numerous SSH exploits over the years that the attacker could try to exploit – especially for SSH1 servers.

Correlations:

http://www.securityfocus.com/archive/75/221281 This posting describes an almost identical scan to port 22 with apparently the same tool.

Evidence of active targeting:

This was probably a scan of a large number of networks, and not specifically targeted to my network. I say this because all of my IP addresses, active or not, were scanned.

Severity:

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

Criticality = 4
These are my publicly available servers.

Severity = 5
SSH exploits can lead to root compromise.

System Countermeasures = 4
SSH was up to date on the servers.

Network Countermeasures = 5
The firewall did not allow connections to port 22 (SSH) from the outside. Also, the firewall did not even allow these probes to reach my servers – they were caught outside the firewall by the IDS sensor.

(4+5) – (4+5) = 0

Defensive Recommendations:

As SSH is often left open to the Internet for remote access to servers, it is very important to keep the SSH daemon patched and up to date. Also, do not run with SSH-1 compatibility enabled, as this opens up many problems.

If possible, do not allow SSH in from the Internet – use some other form of remote access instead (such as a secure VPN tunnel).

Multiple choice test question:

A scan of your hosts with the SYN and FIN flags set and destined for port 22 is likely what?

A) Someone trying to stealthily scan for SSH on your hosts.
B) Someone connecting to your servers with SSH.
C) Someone trying to exploit SSH with a buffer overflow .
D) None of the above

Answer: A.

Detect 5 WEB-IIS Unicode2.pl script

Source of Trace:

This event was recorded on the DMZ directed at one of our publicly available (but private) web servers. That is, the web server required authentication and was not meant for general public use.

Detect was generated by:

The device that triggered this alert was Snort version 1.8.1.

The Snort rule that generated the alert is as follows:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS Unic
ode2.pl script (File permission canonicalization")"; uricontent:"/sensepost.exe"
; flags: A+; nocase; classtype:attempted-recon; sid:989; rev:1;)
```

This rule is basically looking for the string "/sensepost.exe" in the URI (http request).

The alert (as displayed by ACID) is a follows:

```
---------------------------------------------------------------------------
#(1 - 32286) [2001-09-20 11:04:36]  WEB-IIS Unicode2.pl script (File permission canonicalization
IPv4: 63.196.251.182 -> MY.NET.2.153
    hlen=5 TOS=0 dlen=88 ID=23176 flags=0 offset=0 TTL=117 chksum=234
TCP:  port=2045 -> dport: 80  flags=***AP*** seq=3424157827
    ack=67068083 off=5 res=0 win=17520 urp=0 chksum=38974
Payload:  length = 48

000 : 47 45 54 20 2F 73 61 6D 70 6C 65 73 2F 73 65 6E   GET /samples/sen
010 : 73 65 70 6F 73 74 2E 65 78 65 3F 2F 63 2B 64 69   sepost.exe?/c+di
020 : 72 20 48 54 54 50 2F 31 2E 30 0D 0A 0D 0A 0D 0A   r HTTP/1.0.....
```

The interesting fields are as follows:

The source IP address is 63.196.251.182 (adsl-63-196-251-182.dsl.lsan03.pacbell.net), owned by Pac Bell Internet Services (NETBLK-PBI-NET-7) PBI-NET-7. This source IP address is likely a home DSL computer, which may have been directly owned by the attacker, or since it is an always-on broadband connection, it may have been taken over via Trojan from an unknowing owner.

The destination IP address is my server.

The ID of 23176 doesn't stand out as strange, and a quick search of Google for this ID didn't turn anything up. That is, this ID does not appear to be associated with a well known packet crafting tool.

The TTL of 117 is normal, and probably started with an initial TTL of 128, which would likely make this a Windows box.

The source port of 2045 is a typical ephemeral port – especially for a Windows application.

The destination of port 80 is the HTTP port on my web server.

The flags ACK and PSH are set. This means it is an established TCP session (ACK), and the PSH simply means that the application wants the data to be processed immediately and not wait for the buffers to fill.

This all adds up to a normal TCP connection that could easily be made from a web browser, except the payload does not contain any of the extra Accept info or User-Agent info that a web browser often sends along with a request. This leads me to believe that this is more likely to have come from a script rather than a web browser.

The content of the payload is the really interesting part:
GET /samples/sensepost.exe?/c+dir HTTP/1.0

Probability the source address was spoofed:

There is very little chance that the source address was spoofed. This is a reconnaissance scan (just seeing if the DIR will execute in the C drive – not actually doing any damage), and the attacker would want to know if the request completed successfully so he could come back and run more interesting commands. Also, completing a three-way TCP handshake (which is required by the web server before it will accept the GET request) is extremely difficult to do with a forged source IP address. While there are several possibilities for this, including that the attacker could have control of a machine close to the IP address he is spoofing (to sniff sequence numbers off the wire as they pass by) and then use these sequence numbers in forged packets, this is extremely unlikely.

Attack Mechanism:

The interesting thing about this alert is that you have to look at the larger picture to see that this wasn't actually the Unicode2.pl script. While Snort reports this as the Unicode2.pl script (available here http://www.securiteam.com/exploits/Additional_details_about_the_IIS_remote_execution_vulnerability.html called Unicodexecute), this scan was not really the result of this script.

The real Unicode2.pl script works by exploiting the famous Unicode bug in IIS to escape the web directory and execute commands outside the IIS webroot (inetpub). One thing that the Unicode2.pl script did was to copy the binary c:\winnt\system32\cmd.exe to \inetpub\scripts\sensepost.exe. Leaving the binary in the scripts directly opens easy access to controlling the server remotely – even if the Unicode patch is eventually placed on the server. Other Unicode exploits have done similar things, including Code Red copying cmd.exe into \inetpub\scripts\root.exe. (see http://www.europe.f-secure.com/v-descs/bady.shtml).

While it may look like the server was under attack by the Unicode2.pl script, correlation from Snort shows that this was just part of a large scan from this hosts for many common web vulnerabilities. That is, they weren't running the Unicode2.pl against us – they were running a scanner that had a test to see if Unicode2.pl had run successfully against this server some time in the past. If the attacker found sensepost.exe in the scripts directory, then he would have still been able to take control of the server regardless of how many patches had been applied to it.

Correlations:

The real Unicode2.pl script can be found at http://www.securiteam.com/exploits/Additional_details_about_the_IIS_remote_execution_vulnerability.html called Unicodexecute.

Also, here is the correlation of all the tests that this scan did on my server. There are 98 tests total – mostly for IIS, but some meant for Unix (such as the /etc/passwd attempts). I don't recognize the scanning tool itself, but it appears to be similar to an EEYE (Retina) scan or the Cerebus Internet Scanner (a very noisy scanner that attempts to find many dangerous scripts).

This is the larger picture that was required for me to realize that this alert wasn't what it seemed.

#0-(1-32245) [arachNIDS] WEB-IIS ISAPI .ida access 2001-09-20 11:03:44 63.196.251.182:1488 MY.NET.2.153:80 TCP
#1-(1-32251) [CVE] [arachNIDS] WEB-IIS ISAPI .printer access 2001-09-20 11:03:53 63.196.251.182:1544 MY.NET.2.153:80 TCP
#2-(1-32252) [bugtraq] WEB-COLDFUSION startstop DOS access 2001-09-20 11:04:23 63.196.251.182:1586 MY.NET.2.153:80 TCP
#3-(1-32253) [arachNIDS] WEB-CGI websendmail access 2001-09-20 11:04:24 63.196.251.182:1623 MY.NET.2.153:80 TCP
#4-(1-32254) [arachNIDS] WEB-CGI webdriver access 2001-09-20 11:04:26 63.196.251.182:1559 MY.NET.2.153:80 TCP
#5-(1-32255) [CVE] WEB-CGI htmlscript access 2001-09-20 11:04:26 63.196.251.182:1601 MY.NET.2.153:80 TCP
#6-(1-32256) WEB-CGI man.sh access 2001-09-20 11:04:26 63.196.251.182:1602 MY.NET.2.153:80 TCP
#7-(1-32257) [bugtraq] [CVE] WEB-IIS JET VBA access 2001-09-20 11:04:27 63.196.251.182:1690 MY.NET.2.153:80 TCP
#8-(1-32258) WEB-MISC cpshost.dll access 2001-09-20 11:04:27 63.196.251.182:1695 MY.NET.2.153:80 TCP
#9-(1-32259) WEB-MISC cpshost.dll access 2001-09-20 11:04:27 63.196.251.182:1698 MY.NET.2.153:80 TCP
#10-(1-32260) [CVE] WEB-CGI wguest.exe access 2001-09-20 11:04:27 63.196.251.182:1714 MY.NET.2.153:80 TCP
#11-(1-32261) [bugtraq] WEB-COLDFUSION evaluate.cfm access 2001-09-20 11:04:29 63.196.251.182:1742 MY.NET.2.153:80 TCP
#12-(1-32262) [bugtraq] WEB-COLDFUSION snippets attempt 2001-09-20 11:04:29 63.196.251.182:1750 MY.NET.2.153:80 TCP
#13-(1-32263) WEB-COLDFUSION administrator access 2001-09-20 11:04:30 63.196.251.182:1665 MY.NET.2.153:80 TCP
#14-(1-32264) [bugtraq] WEB-COLDFUSION parks 2001-09-20 11:04:30 63.196.251.182:1753 MY.NET.2.153:80 TCP
#15-(1-32265) [arachNIDS] WEB-CGI php access 2001-09-20 11:04:30 63.196.251.182:1658 MY.NET.2.153:80 TCP
#16-(1-32266) [bugtraq] WEB-IIS site server config access 2001-09-20 11:04:30 63.196.251.182:1667 MY.NET.2.153:80 TCP
#17-(1-32267) WEB-FRONTPAGE fpadmcgi.exe access 2001-09-20 11:04:30 63.196.251.182:1671 MY.NET.2.153:80 TCP

#18-(1-32268) [bugtraq] [CVE] WEB-IIS JET VBA access 2001-09-20 11:04:30 63.196.251.182:1675 MY.NET.2.153:80 TCP
#19-(1-32269) [bugtraq] WEB-COLDFUSION snippets attempt attempt 2001-09-20 11:04:30 63.196.251.182:1762 MY.NET.2.153:80 TCP
#20-(1-32270) WEB-MISC /etc/passwd 2001-09-20 11:04:30 63.196.251.182:1785 MY.NET.2.153:80 TCP
#21-(1-32271) WEB-COLDFUSION administrator access 2001-09-20 11:04:33 63.196.251.182:1758 MY.NET.2.153:80 TCP
#22-(1-32272) [arachNIDS] MISC PCCS mysql database admin tool 2001-09-20 11:04:33 63.196.251.182:1844 MY.NET.2.153:80 TCP
#23-(1-32273) WEB-MISC repost.asp access 2001-09-20 11:04:34 63.196.251.182:1855 MY.NET.2.153:80 TCP
#24-(1-32274) WEB-MISC /etc/passwd 2001-09-20 11:04:34 63.196.251.182:1859 MY.NET.2.153:80 TCP
#25-(1-32275) WEB-MISC /etc/passwd 2001-09-20 11:04:34 63.196.251.182:1872 MY.NET.2.153:80 TCP
#26-(1-32276) WEB-MISC /etc/passwd 2001-09-20 11:04:34 63.196.251.182:1874 MY.NET.2.153:80 TCP
#27-(1-32277) [arachNIDS] WEB-CGI whoisraw access 2001-09-20 11:04:34 63.196.251.182:1902 MY.NET.2.153:80 TCP
#28-(1-32278) [bugtraq] [CVE] WEB-COLDFUSION expeval access 2001-09-20 11:04:34 63.196.251.182:1919 MY.NET.2.153:80 TCP
#29-(1-32279) WEB-IIS SAM Attempt 2001-09-20 11:04:34 63.196.251.182:1921 MY.NET.2.153:80 TCP
#30-(1-32280) [bugtraq] [CVE] WEB-COLDFUSION expeval access 2001-09-20 11:04:34 63.196.251.182:1928 MY.NET.2.153:80 TCP
#31-(1-32281) [arachNIDS] WEB-MISC Phorum code access 2001-09-20 11:04:35 63.196.251.182:1954 MY.NET.2.153:80 TCP
#32-(1-32282) [bugtraq] WEB-MISC Talentsoft Web+ exploit access 2001-09-20 11:04:35 63.196.251.182:1986 MY.NET.2.153:80 TCP
#33-(1-32283) [arachNIDS] [CVE] WEB-CGI wrap access 2001-09-20 11:04:36 63.196.251.182:1669 MY.NET.2.153:80 TCP
#34-(1-32285) [bugtraq] [CVE] WEB-IIS iisadmpwd attempt 2001-09-20 11:04:36 63.196.251.182:2036 MY.NET.2.153:80 TCP
#35-(1-32286) WEB-IIS Unicode2.pl script (File permission canonicalization 2001-09-20 11:04:36 63.196.251.182:2045 MY.NET.2.153:80 TCP
#36-(1-32287) WEB-MISC shopping cart access access 2001-09-20 11:04:36 63.196.251.182:1833 MY.NET.2.153:80 TCP
#37-(1-32289) WEB-MISC VirusWall FtpSave access 2001-09-20 11:04:37 63.196.251.182:2112 MY.NET.2.153:80 TCP
#38-(1-32290) WEB-IIS admin access 2001-09-20 11:04:37 63.196.251.182:1732 MY.NET.2.153:80 TCP
#39-(1-32291) WEB-MISC /etc/passwd 2001-09-20 11:04:37 63.196.251.182:2124 MY.NET.2.153:80 TCP
#40-(1-32292) [arachNIDS] WEB-CGI wwwboard passwd access 2001-09-20 11:04:38 63.196.251.182:2006 MY.NET.2.153:80 TCP
#41-(1-32293) [arachNIDS] WEB-CGI wwwboard passwd access 2001-09-20 11:04:38 63.196.251.182:2140 MY.NET.2.153:80 TCP
#42-(1-32294) WEB-MISC /etc/passwd 2001-09-20 11:04:38 63.196.251.182:2018 MY.NET.2.153:80 TCP
#43-(1-32295) WEB-MISC VirusWall FtpSaveCVP access 2001-09-20 11:04:39 63.196.251.182:2073 MY.NET.2.153:80 TCP
#44-(1-32296) [arachNIDS] WEB-IIS ISAPI .idq access 2001-09-20 11:04:39 63.196.251.182:2100 MY.NET.2.153:80 TCP
#45-(1-32300) WEB-IIS Unicode2.pl script (File permission canonicalization 2001-09-20 11:04:41 63.196.251.182:2177 MY.NET.2.153:80 TCP
#46-(1-32301) WEB-IIS Unicode2.pl script (File permission canonicalization 2001-09-20 11:04:41 63.196.251.182:2183 MY.NET.2.153:80 TCP
#47-(1-32311) [CVE] WEB-MISC /cgi-bin/jj attempt 2001-09-20 11:07:49 63.196.251.182:2316 MY.NET.121.217:80 TCP
#48-(1-32312) [bugtraq] WEB-IIS search97.vts 2001-09-20 11:07:49 63.196.251.182:2332 MY.NET.121.217:80 TCP
#49-(1-32313) WEB-MISC nc.exe attempt 2001-09-20 11:07:50 63.196.251.182:2336 MY.NET.121.217:80 TCP
#50-(1-32314) [arachNIDS] [CVE] WEB-MISC handler access 2001-09-20 11:07:50 63.196.251.182:2355 MY.NET.121.217:80 TCP
#51-(1-32315) WEB-CGI man.sh access 2001-09-20 11:07:50 63.196.251.182:2363 MY.NET.121.217:80 TCP
#52-(1-32316) [CVE] WEB-MISC count.cgi access 2001-09-20 11:07:50 63.196.251.182:2365 MY.NET.121.217:80 TCP
#53-(1-32317) WEB-CGI testcounter.pl access 2001-09-20 11:07:50 63.196.251.182:2373 MY.NET.121.217:80 TCP
#54-(1-32318) [arachNIDS] [CVE] WEB-CGI phf access 2001-09-20 11:07:51 63.196.251.182:2376 MY.NET.121.217:80 TCP
#55-(1-32319) [CVE] [arachNIDS] WEB-CGI test-cgi access 2001-09-20 11:07:51 63.196.251.182:2381 MY.NET.121.217:80 TCP
#56-(1-32320) [arachNIDS] WEB-CGI webgais access 2001-09-20 11:07:51 63.196.251.182:2383 MY.NET.121.217:80 TCP
#57-(1-32321) [arachNIDS] WEB-CGI websendmail access 2001-09-20 11:07:51 63.196.251.182:2388 MY.NET.121.217:80 TCP
#58-(1-32322) WEB-CGI www-sql access 2001-09-20 11:07:52 63.196.251.182:2389 MY.NET.121.217:80 TCP
#59-(1-32323) WEB-IIS admin access 2001-09-20 11:07:52 63.196.251.182:2399 MY.NET.121.217:80 TCP
#60-(1-32324) [arachNIDS] WEB-CGI webgais access 2001-09-20 11:07:52 63.196.251.182:2407 MY.NET.121.217:80 TCP
#61-(1-32325) [bugtraq] WEB-MISC counter.exe access 2001-09-20 11:07:53 63.196.251.182:2445 MY.NET.121.217:80 TCP
#62-(1-32326) WEB-MISC postinfo.asp access 2001-09-20 11:07:53 63.196.251.182:2447 MY.NET.121.217:80 TCP
#63-(1-32327) [CVE] WEB-CGI aglimpse access 2001-09-20 11:07:53 63.196.251.182:2455 MY.NET.121.217:80 TCP
#64-(1-32328) [bugtraq] WEB-MISC counter.exe access 2001-09-20 11:07:54 63.196.251.182:2474 MY.NET.121.217:80 TCP
#65-(1-32329) WEB-IIS admin access 2001-09-20 11:07:54 63.196.251.182:2495 MY.NET.121.217:80 TCP
#66-(1-32330) [CVE] [arachNIDS] WEB-CGI formmail access 2001-09-20 11:07:55 63.196.251.182:2508 MY.NET.121.217:80 TCP
#67-(1-32331) WEB-IIS showcode access 2001-09-20 11:07:55 63.196.251.182:2518 MY.NET.121.217:80 TCP
#68-(1-32332) WEB-MISC shopping cart access access 2001-09-20 11:07:55 63.196.251.182:2531 MY.NET.121.217:80 TCP
#69-(1-32333) WEB-MISC /etc/passwd 2001-09-20 11:07:56 63.196.251.182:2556 MY.NET.121.217:80 TCP
#70-(1-32334) WEB-IIS showcode access 2001-09-20 11:07:56 63.196.251.182:2563 MY.NET.121.217:80 TCP
#71-(1-32335) WEB-MISC /etc/passwd 2001-09-20 11:07:57 63.196.251.182:2593 MY.NET.121.217:80 TCP
#72-(1-32336) WEB-MISC /etc/passwd 2001-09-20 11:07:58 63.196.251.182:2597 MY.NET.121.217:80 TCP
#73-(1-32337) WEB-MISC repost.asp access 2001-09-20 11:07:58 63.196.251.182:2601 MY.NET.121.217:80 TCP
#74-(1-32338) [bugtraq] WEB-COLDFUSION getfile.cfm access 2001-09-20 11:07:58 63.196.251.182:2509 MY.NET.121.217:80 TCP

#75-(1-32339) WEB-MISC repost.asp access 2001-09-20 11:07:58 63.196.251.182:2610 MY.NET.121.217:80 TCP
#76-(1-32340) [CVE] WEB-CGI campas access 2001-09-20 11:08:00 63.196.251.182:2623 MY.NET.121.217:80 TCP
#77-(1-32341) WEB-MISC order.log access 2001-09-20 11:08:02 63.196.251.182:2655 MY.NET.121.217:80 TCP
#78-(1-32342) [arachNIDS] WEB-CGI whoisraw access 2001-09-20 11:08:02 63.196.251.182:2663 MY.NET.121.217:80 TCP
#79-(1-32343) WEB-IIS Unicode2.pl script (File permission canonicalization 2001-09-20 11:08:02 63.196.251.182:2680 MY.NET.121.217:80 TCP
#80-(1-32344) WEB-IIS Unicode2.pl script (File permission canonicalization 2001-09-20 11:08:04 63.196.251.182:2720 MY.NET.121.217:80 TCP
#81-(1-32345) [arachNIDS] WEB-MISC Phorum code access 2001-09-20 11:08:04 63.196.251.182:2757 MY.NET.121.217:80 TCP
#82-(1-32346) WEB-CGI files.pl access 2001-09-20 11:08:05 63.196.251.182:2766 MY.NET.121.217:80 TCP
#83-(1-32348) WEB-MISC VirusWall FtpSaveCVP access 2001-09-20 11:08:06 63.196.251.182:2792 MY.NET.121.217:80 TCP
#84-(1-32349) WEB-FRONTPAGE fpcount.exe access 2001-09-20 11:08:06 63.196.251.182:2799 MY.NET.121.217:80 TCP
#85-(1-32350) [CVE] [bugtraq] WEB-CGI cvsweb.cgi access 2001-09-20 11:08:06 63.196.251.182:2826 MY.NET.121.217:80 TCP
#86-(1-32351) [CVE] WEB-IIS .... access 2001-09-20 11:08:08 63.196.251.182:2848 MY.NET.121.217:80 TCP
#87-(1-32352) WEB-MISC /etc/passwd 2001-09-20 11:08:08 63.196.251.182:2850 MY.NET.121.217:80 TCP
#88-(1-32353) [arachNIDS] WEB-IIS ISAPI .idq access 2001-09-20 11:08:09 63.196.251.182:2866 MY.NET.121.217:80 TCP
#89-(1-32354) WEB-IIS SAM Attempt 2001-09-20 11:08:09 63.196.251.182:2869 MY.NET.121.217:80 TCP
#90-(1-32355) WEB-MISC /etc/passwd 2001-09-20 11:08:09 63.196.251.182:2881 MY.NET.121.217:80 TCP
#91-(1-32358) [CVE] WEB-IIS .... access 2001-09-20 11:08:12 63.196.251.182:2870 MY.NET.121.217:80 TCP
#92-(1-32359) [CVE] WEB-IIS .... access 2001-09-20 11:08:12 63.196.251.182:2899 MY.NET.121.217:80 TCP
#93-(1-32361) WEB-IIS Unicode2.pl script (File permission canonicalization 2001-09-20 11:08:16 63.196.251.182:2913 MY.NET.121.217:80 TCP
#94-(1-32363) WEB-IIS Unicode2.pl script (File permission canonicalization 2001-09-20 11:08:20 63.196.251.182:2924 MY.NET.121.217:80 TCP
#95-(1-32364) WEB-IIS Unicode2.pl script (File permission canonicalization 2001-09-20 11:08:20 63.196.251.182:2925 MY.NET.121.217:80 TCP
#96-(1-32365) WEB-IIS Unicode2.pl script (File permission canonicalization 2001-09-20 11:08:21 63.196.251.182:2931 MY.NET.121.217:80 TCP
#97-(1-32366) WEB-IIS Unicode2.pl script (File permission canonicalization 2001-09-20 11:08:22 63.196.251.182:2937 MY.NET.121.217:80 TCP
#98-(1-32367) WEB-IIS Unicode2.pl script (File permission canonicalization 2001-09-20 11:08:22 63.196.251.182:2943 MY.NET.121.217:80 TCP

Evidence of active targeting:

This was actively targeted in the sense that the attacker only scanned this one, private (but available to the Internet) server. Even though our public servers are in the same class 'C' netblock, the attacker appears to have made no attempt to scan them. This was probably due to earlier reconnaissance by the attacker (possibly from a different IP address) to determine that this server was our only IIS server (the other publicly available servers were all Unix web servers). Since he clearly targeted our only IIS server with a tool that scanned for mostly IIS problems, this attack was targeted. Apparently, this attacker only knew how to attack IIS (which is one reason our other web servers are not running IIS).

Severity:

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

Criticality = 4
This is an important application server.

Lethality = 4
If successful, the exploit would allow full remote control of the server (as the IIS user). If the server is not patched correctly, escalating the privileges to the Administrator is possible.

System Countermeasures = 4
All IIS patches plus NT operating system patches were applied. All of Microsoft's recommendations for running a secure IIS server had been followed (see http://www.fulgan.com/delphi/CheckList.htm) This included such things as removing unneeded sample scripts, removing all handlers except ASP (such as .HRT, .IDQ, etc), placing the webroot on a different physical drive than the system root, etc. Also, there were no executable directories on this web server (not /scripts, etc), so even if the server had never been patched for Unicode, it would not have been vulnerable to it.

Network Countermeasures = 4
The Webserver was in a protected DMZ, and it could not be connected to on any port other than 80, and could not make any connection out to the Internet. Not being able to connect to the Internet helps to prevent privilege escalation programs from being uploaded onto the server to give the attacker Administrator privileges. Also, since the firewall only allowes connection to port 80, the attacker could not exploit other services on the server, or leave NetCat (or similar network servers) running on an unused port to connect back to.

(4 + 4) – (4 + 4) = 0

Defensive Recommendations:

Place IIS servers behind a firewall that only allows connections to the web server port. This prevents connections to other dangerous services, such as RPC or Net Bios. Also, don't allow the web servers to initiate connections out to the Internet. Not being able to connect to the Internet helps to prevent privilege escalation programs from being uploaded onto the server to give the attacker Administrator privileges. Also, follow all of the recommendations in the Microsoft guide "Microsoft Internet Information Server 4.0 Security Checklist", available here (http://www.fulgan.com/delphi/CheckList.htm)

Multiple choice test question:

If you see /scripts/sensepost.exe in a web request to your IIS server, it is possibly

A) Someone is scanning to see if you have been previously exploited by the Unicode2.pl script.
B) Someone is trying to use the Unicode bug (with the script Unicode2.pl) to compromise your server.
C) Someone is trying to post malicious code to your web server using sensepost.exe.
D) Either A or B

Answer: D.

Assignment 3 - "Analyze This" Scenario

List of files Analyzed:

alert.020125.gz
alert.020126.gz
alert.020127.gz
alert.020128.gz
alert.020129.gz

scans_021025.gz
scans_021026.gz
scans_021027.gz
scans_021028.gz
scans_021029.gz

oos_Jan.25.2002.gz
oos_Jan.26.2002.gz
oos_Jan.27.2002.gz
oos_Jan.28.2002.gz
oos_Jan.29.2002.gz

Overview of analysis:

This appears to be an open college network, without a strong perimeter defense (restrictive firewall). Without a strong perimeter defense, this network is heavily scanned and mapped by outside IP addresses. As the data provided does not include the raw packet dumps, it is difficult to determine how many of the attacks have actually succeeded. It does appear, however, that several of the machines may have the Red Worm and the Back Orifice Trojan – as well as the My Party virus.

Defensive Recommendations:

Because it is an open network, it is critically important for this site to properly configure and maintain its machines. They should follow SANS or CERT guidelines on hardening machines, and only run the minimum services required.

Also, intrusion detection is important at a site like this, as it can alert the staff to machines that appear to be compromised, and which are either scanning for new hosts, participating in Denial of Services attacks, or communicating to Trojan servers.

If at all possible, identify servers that are mission critical, and place those in on a network this is protected by a restrictive firewall. The use of host based intrusion detection should also be considered for these critical systems.

Relationships and possible compromises.

These are some relationships between machines and some possibly compromised machines.

The host MY.NET.5.96 is heavily scanned for web vulnerabilities from outside the network. This appears to be a public web server, or at least one that is well known to attackers. Of particular note with this host, is that in addition to all the scanning, this host also registers 5 alerts for possible myserver activity, which is a Trojan program. This opens the possibility that some of the attacks have been successful.

The host MY.NET.70.177 has almost 12,000 alerts attributed to it for polling SNMP. This is likely a network management station, such as HP Openview.

The host MY.NET.150.198 received over 55,000 connections (from 103 sources) to port 515, which is typically used for printing. This port has had problems with buffer overflows in Unix recently, but the traffic pattern to this host makes it appear to be a large print server (and not the target of random scanning).

The following servers all had more that 1000 alerts total, and included the IIS Unicode attack. Many of these servers also register SMB wildcard attacks (Windows share enumeration) and connections to printer servers. These are likely internal servers, that possibly have one or more services running (web, SMB, printer).

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
MY.NET.153.123	5003	5631	53	55
MY.NET.153.185	3052	3417	50	52
MY.NET.153.187	2697	2697	42	42
MY.NET.153.137	2634	3063	22	24
MY.NET.153.119	2588	8248	24	27
MY.NET.153.122	2140	4412	26	31
MY.NET.153.184	1971	1972	21	22
MY.NET.151.108	1926	1927	11	12
MY.NET.153.121	1780	2365	15	21

MY.NET.153.115	1587	3277	38	40
MY.NET.153.113	1264	3617	42	45
MY.NET.150.165	1242	1645	31	51
MY.NET.153.110	1234	2657	28	33
MY.NET.153.171	1197	1257	45	45
MY.NET.153.108	1120	1595	39	42
MY.NET.153.144	1013	1053	35	37

These are the servers that were either the source or the destination of the Back Orifice alert. This is a very serious Trojan, and these hosts should be checked.

Sources triggering this attack signature

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
MY.NET.6.48	3	685	2	67
MY.NET.6.49	2	690	2	73
MY.NET.6.52	2	573	2	75
216.106.172.149	1	23	1	3

Destinations receiving this attack signature

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
MY.NET.153.184	2	17	1	5
MY.NET.153.193	1	72	1	8
MY.NET.153.197	1	53	1	3
MY.NET.152.181	1	317	1	26
MY.NET.153.208	1	9	1	2
MY.NET.153.161	1	5	1	4
MY.NET.153.166	1	39	1	6

The host MY.NET.60.43 is apparently an NTP server, as it had several hundred thousand connections to the NTP port.

The following hosts should be checked, as they registered the alert EXPLOIT x86 NOOP. This could mean that a buffer overflow exploit has been run against them. This seems especially valid (less likely to be false positive), as all of the source IP addresses for the signature were external.

Destinations	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
MY.NET.150.143	12	15	1	3
MY.NET.153.202	4	45	1	5
MY.NET.88.163	1	742	1	2
MY.NET.153.118	1	1	1	1
MY.NET.153.208	1	9	1	2
MY.NET.153.142	1	30	1	10
MY.NET.153.184	1	17	1	5
MY.NET.5.249	1	1482	1	2

There are numerous hosts running Instant Message programs and file swapping programs. If these are allowed, the Snort rules that log them should probably be removed to not clutter up the logs. If they are not allowed, host owners should be contacted, or firewall changes should be made to block them.

There are also numerous hosts reporting Red Worm and the My Party virus. An attempt to refine the Snort rule should be made to make it less prone to false positives. As Red Worm and My Party are both serious problems, these should be investigated further.

The host MY.NET.6.45 is connecting form port 7000 to port 7001 on various internal machines. This appears to be an AFS file server. AFS is a distributed file system similar to NFS.

Top Talkers (for all alert data).

Here is the complete list of alerts from the entire 5 day period, prioritized by number of alerts (in reverse order).

Rank	Total # Alerts	**Source IP**	# Signatures triggered	Destinations involved

rank #1	11930 alerts	MY.NET.70.177	2 signatures	(26 destination IPs)
rank #2	10740 alerts	63.250.209.34	1 signatures	MY.NET.151.63, 255.255.153.210
rank #3	8248 alerts	MY.NET.153.119	4 signatures	(27 destination IPs)
rank #4	7748 alerts	MY.NET.153.114	4 signatures	(25 destination IPs)
rank #5	7246 alerts	63.250.210.50	1 signatures	MY.NET.151.63
rank #6	5631 alerts	MY.NET.153.123	3 signatures	(55 destination IPs)
rank #7	5149 alerts	MY.NET.153.111	3 signatures	(24 destination IPs)
rank #8	4970 alerts	63.250.208.34	1 signatures	255.255.151.63
rank #9	4621 alerts	MY.NET.153.118	3 signatures	(18 destination IPs)
rank #10	4412 alerts	MY.NET.153.122	6 signatures	(31 destination IPs)

Selected External IP addresses:

Note: WHOIS queried through Sam Spade web site. http://www.samspade.org

IP address #1 -- 64.226.244.176.

I chose this IP address because it was the source of numerous attempts to exploit the .printer isapi. This went beyond simply scanning, and there were actual attempts at the buffer overflow. This IP address comes from Interland, which is an ISP and web hosting company.

This host had the following alerts:

1 instance of WEB-IIS 5 .printer isapi
1 instance of EXPLOIT x86 NOOP
6 instances of WEB-IIS 5 Printer-beavuh

Trying whois -h whois.arin.net 64.226.244.176

Interland (NETBLK-INTERLAND-5)
303 Peachtree Suite 500
Atlanta, GA 30303
US

Netname: INTERLAND-5
Netblock: 64.224.0.0 - 64.227.255.255
Maintainer: INTD

Coordinator:
Interland, Inc. (II27-ARIN) asnadmin@interland.com
404-720-3720

IP address #2 – 64.152.108.141.

This IP address was responsible for numerous scans, and was also involved in the possible Red Worm activity. This IP address comes from Level3, which is an ISP and web hosting company.

This host had the following alerts:

1 instance of Attempted Sun RPC high port access
1 instance of EXPLOIT x86 NOOP
2 instances of RPC udp traffic contains bin sh
4 instances of TFTP - External UDP connection to internal tftp server
36 instances of EXPLOIT NTPDX buffer overflow
278 instances of High port 65535 udp - possible Red Worm - traffic

Level 3 Communications, Inc. (NETBLK-LC-ORG-ARIN) LC-ORG-ARIN
64.152.0.0 - 64.159.255.255
Streaming Media Corporation (NETBLK-NETBLK-STRM9) NETBLK-STRM9
64.152.108.0 - 64.152.108.255

IP address #3 – 68.54.201.254

This IP address was attempting to exploit the Unicode attack (directory traversal). This IP address comes from Comcast Cable, so it probably belongs to a home broadband user.

This host had the following alerts:

1 instance of WEB-CGI scriptalias access
2 instances of WEB-MISC http directory traversal
7 instances of spp_http_decode: IIS Unicode attack detected

Comcast Cable Communications, Inc. (NETBLK-JUMPSTART-1) JUMPSTART-1
68.32.0.0 - 68.63.255.255
Comcast Cable Communications, Inc. (NETBLK-JUMPSTART-BALTIMORE-A) JUMPSTART-BALTIMORE-A
68.54.80.0 - 68.55.255.255

IP address #4 – 63.250.209.34

This IP address was responsible for 10,740 large UDP packets. This is apparently some form of streaming media, as it comes from Yahoo! Broadcast services.

This host had the following alerts:

Yahoo! Broadcast Services, Inc. (NETBLK-NETBLK2-YAHOOBS)
2914 Taylor St
Dallas, TX 75226
US

Netname: NETBLK2-YAHOOBS
Netblock: 63.250.192.0 - 63.250.223.255
Maintainer: YAHO

Coordinator:
Bonin, Troy (TB501-ARIN) netops@broadcast.com
214.782.4278 ext. 2278

IP address #5 – 24.162.192.226.

This IP address was searching for proxy servers – possibly to use in hiding his/her web surfing and hacking. The IP address comes from Road Runner, which is an ISP.

This host had the following alerts:

1 instance of INFO - Possible Squid Scan

2 instances of SCAN Proxy attempt

ServiceCo LLC - Road Runner (NET-ROAD-RUNNER-5)
13241 Woodland Park Road
Herndon, VA 20171
US

Netname: ROAD-RUNNER-5
Netblock: 24.160.0.0 - 24.170.127.255
Maintainer: SCRR

Coordinator:
ServiceCo LLC (ZS30-ARIN) abuse@rr.com
1-703-345-3416

Analysis Process:

The input files were analyzed using SnortSnarf v020126.1. See http://www.sicicondefense.com

List of Alerts:

Signature	# Alerts	# Sources	# Dests
MISC Large ICMP Packet	1	1	1
WEB-MISC webdav search access [arachNIDS]	1	1	1
MISC PCAnywhere Startup	1	1	1
Tiny Fragments - Possible Hostile Activity	1	1	1
WEB-IIS 5 .printer isapi	1	1	1
BACKDOOR NetMetro File List	1	1	1

ICMP Echo Request Delphi-Piette Windows	1	1	1
WEB-IIS Unauthorized IP Access Attempt	1	1	1
RPC udp traffic contains bin sh	2	1	1
WEB-MISC http directory traversal [arachNIDS]	2	1	1
WEB-IIS encoding access [arachNIDS]	2	2	1
ICMP Address Mask Request	2	1	1
ICMP SRC and DST outside network	2	1	1
Watchlist 000222 255-NCFC	2	1	1
INFO - Possible Squid Scan	2	2	2
TFTP - External UDP connection to internal tftp server	4	1	1
EXPLOIT x86 setgid 0	4	4	4
Probable NMAP fingerprint attempt	4	1	1
ICMP Echo Request CyberKit 2.2 Windows	5	1	3
EXPLOIT x86 setuid 0	5	3	4
ICMP Address Mask Reply	6	2	2
SUNRPC highport access!	6	2	1
Queso fingerprint	6	3	2
WEB-IIS 5 Printer-beavuh	6	1	6
WEB-CGI formmail access [CVE] [arachNIDS]	7	7	2
High port 65535 tcp - possible Red Worm - traffic	8	2	2
Back Orifice	8	4	7
MISC source port 53 to <1024	8	8	3
Port 55850 tcp - Possible myserver activity - ref. 010313-1	10	4	4
TCP SRC and DST outside network	10	3	7
INFO - ICQ Access	13	1	3
WEB-MISC whisker head	16	1	1
Incomplete Packet Fragments Discarded	16	10	3
Russia Dynamo - SANS Flash 28-jul-00	16	2	2

ICMP Destination Unreachable (Protocol Unreachable)	18	4	5
Attempted Sun RPC high port access	19	9	12
EXPLOIT x86 NOOP	22	8	8
ICMP traceroute	24	14	7
WEB-MISC 403 Forbidden	32	3	15
WEB-MISC compaq nsight directory traversal [CVE] [arachNIDS]	37	6	6
Possible trojan server activity	38	10	10
SCAN Synscan Portscan ID 19104	41	41	4
INFO FTP anonymous FTP	46	2	20
INFO Outbound GNUTella Connect accept	48	48	2
ICMP Echo Request Windows	51	14	7
NMAP TCP ping!	58	9	4
MISC traceroute	58	11	4
WEB-CGI scriptalias access	60	8	1
WEB-MISC Attempt to execute cmd	77	10	4
INFO Possible IRC Access	82	14	21
SCAN Proxy attempt	88	13	7
EXPLOIT NTPDX buffer overflow	103	12	8
ICMP Destination Unreachable (Communication Administratively Prohibited)	140	1	1
WEB-FRONTPAGE _vti_rpc access	161	55	1
WEB-IIS _vti_inf access	172	57	1
ICMP Echo Request BSDtype	421	5	7
Null scan!	473	93	5
255PARTY - Possible My Party infection	525	4	1
FTP DoS ftpd globbing	531	3	1
ICMP Fragment Reassembly Time Exceeded	763	27	33
WEB-IIS view source via translate header [arachNIDS]	842	18	1

ICMP Echo Request Nmap or HPING2	1312	60	5
ICMP Router Selection	1622	139	1
Watchlist 000220 IL-ISDN255-990517	1807	34	6
spp_http_decode: CGI Null Byte attack detected	2502	8	14
High port 65535 udp - possible Red Worm - traffic	3995	85	134
INFO MSN IM Chat data	5445	77	78
ICMP Echo Request L3retriever Ping	8599	96	14
SMB Name Wildcard	17454	172	181
MISC Large UDP Packet [arachNIDS]	23908	9	7
SNMP public access	24042	17	143
spp_http_decode: IIS Unicode attack detected	53179	123	568
connect to 515 from inside	55269	103	1

Here is a brief description of each alert:

MISC Large ICMP Packet

ICMP packets are typically small, as they carry very little data. A large ICMP packet can be a sign of an attempt to flood the network (denial of service), the famous ping of death, or harmless WAN circuit testing. Since there is only one, this does not appear to be a problem.

WEB-MISC webdav search access

Webdav is an extension to HTTP found on IIS5 web servers. There was a denial of service attack against this service. The source IP address (12.91.164.96) was a host doing extensive IIS scanning against the target IP, so this should be considered malicious.

Defensive Recommendation:
First, make sure all appropriate patches are applied to any publicly available web server – especially IIS. Second, the source IP address could probably be blocked, as the attacker appears to be scanning for IIS exploits.

MISC PCAnywhere Startup

This is showing that the remote control program PCAnywhere is starting up. As the source IP address was outside the network, this could be a problem. The application gives complete control of a Windows computer.

Defensive Recommendation:
Verify that PCAnywhere is an allowed application. If it is not, then remove the application from the destination PC, and block the ports the applications uses at the firewall.

Tiny Fragments - Possible Hostile Activity

Tiny fragments are not normal traffic, but in this case, since there is only one, it can be safely ignored. Tiny fragments are often used to try to get past simple packet filtering firewalls, or as a denial of service attack. Certain versions of CheckPoint firewall-1 are vulnerable to a denial of service if they are subjected to a sustained stream of tiny fragments.

Correlation:
http://www.giac.org/practical/Garreth_jeremiah_GCIA.zip

WEB-IIS 5 .printer isapi

This is a serious hole, as it allows remote administration access to a server running unpatched IIS 5.0 via a buffer overflow in the .printer isapi (file type handler). The source IP address, 64.226.244.176 also sent a x86 NOOP attack against the same IP address, so there is a possibility that the attacker tried to exploit this buffer overflow.

Defensive Recommendation:
Apply all patches for IIS, and also remove the .printer isapi from the system.

BACKDOOR NetMetro File List

This is a Trojan horse that allows an attacker to take control of the PC. I have found this to be a common false alert from web servers, as the rule is looking for a connection from any port internally to port 5032 on the outside (plus the data |2D 2D|). When this alert comes from port 80, it is probably just the HTTP reply from a web browser that has chosen the ephemeral port 5032 from which to start.

ICMP Echo Request Delphi-Piette Windows

According to this posting on Incidents http://www.securityfocus.com/archive/75/59075, this is simply someone using the Delphi-Piette ICMP library from Borland. As there is only one attempt at this, it is probably not serious.

WEB-IIS Unauthorized IP Access Attempt

This is an IP address that is not allowed to access a private URL on an IIS server attempting to do so. The error that the server returns triggers the alert. The destination IP address for this alert, 130.226.143.97, also has two 403 errors, which are forbidden URL accesses. This may be someone attempting to access administrative or other private parts of the web server.

RPC udp traffic contains bin sh

/bin/sh is code that is used in Unix to spawn an interactive shell. As this alert comes from a host that has 322 other alerts (64.152.108.141), it most likely is an attempt to start a shell through some type of RPC exploit. This is definitely one to check on further.

Defensive Recommendations:

If at all possible, block outside access to RPC services – they are notoriously dangerous. Also, block this particular source IP address, as it is attempting (and possibly succeeding) at numerous exploit attempts.

WEB-MISC http directory traversal

This is someone tying to add '..\\' to a URL. This is an attempt to get out of the Webserver root, to see or execute files that the person is not supposed to. The source IP, 68.54.201.254, also has several Unicode alerts. Unicode is often used as part of a directory traversal for IIS web servers.

WEB-IIS encoding access

The classification is "access to a potentially vulnerable web application" from http://www.snort.org/snort-db/sid.html?id=1010

ICMP Address Mask Request

This is someone requesting the address mask of a host or router. This information can be used by an attacker to help map out a network.

ICMP SRC and DST outside network

These are unusual packets, and without more information, it is difficult to categorize them. Some possibilities include that of internal hosts spoofing external addresses, router or host misconfiguration, VPN or IPSEC tunnels that bridge distant networks to this network, and laptop computers that travel between different networks.

Correlation (and reference):
http://www.giac.org/practical/Garreth_jeremiah_GCIA.zip

Watchlist 000222 255-NCFC

This is watching for any activity from the network NCFC (The Computer Network Center Chinese Academy of Sciences).

Correlation (and reference):
http://www.sans.org/y2k/practical/Mike_Bell_GCIA.doc

INFO - Possible Squid Scan

This is looking for traffic to port 3128, which is the port normally used by Squid.

TFTP - External UDP connection to internal tftp server

This is from an IP address that has 344 alerts over these 5 days (64.152.108.141). This attacker is looking for TFTP servers, as they don't require passwords. Once a TFTP server is found, the attacker may be able to download router configurations or other useful configuration files from it.

EXPLOIT x86 setgid 0

This is an attempt by a program trying to set its group to zero, which would place it in the root group. This could be legitimate (like sudo), or a hacking tool trying to elevate its privileges.

Probable NMAP fingerprint attempt

This is someone attempting to use NMAP to determine the type of operating system that is in use.

ICMP Echo Request CyberKit 2.2 Windows

This is someone using a hacker tool, CyberKit 2.2 to scan the network. As the source IP address is internal, and the destination address are all external, it would be good to check this internal host to see if it has been compromised.

EXPLOIT x86 setuid 0

This is an attempt at a program trying to set its effective ID to zero, which would give it root access. This could be legitimate (like sudo), or a hacking tool trying to elevate its privileges.

ICMP Address Mask Reply

This is similar to the request, except it is the reply to the request.

SUNRPC highport access!

There are multiple sources for this, both internal and external. The internal hosts could be legitimate (i.e. NFS, NIS, etc), but the attempts from outside are probably scanning for well-known RPC program vulnerabilities.

Correlation:
http://www.sans.org/y2k/practical/Andy_Siske_GCIA.htm

Queso fingerprint

Queso is a tool like NMAP for determining the operating system of the remote computer. It does this by setting strange flags in the TCP header, and seeing how the computer responds.

Correlations:
http://www.giac.org/practical/Garreth_jeremiah_GCIA.zip

References:

http://www.giac.org/practical/Garreth_jeremiah_GCIA.zip
CVE Candidate Reference
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0454

WEB-IIS 5 Printer-beavuh

This is an exploit by Dark Spyrit to the .printer exploit documented earlier.

References:
http://rr.sans.org/win2000/5threats.php

WEB-CGI formmail access

This is someone scanning for formmail.cgi on the server. In old versions of this script, the source email address was not checked, so SPAM could be sent from the server that didn't lead back to the actual sender. Also, there is a hole that can allow remote execution of commands.

References:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0172

High port 65535 tcp - possible Red Worm – traffic

This alert is triggered when traffic is coming from or going to port 65535. This is the highest legal TCP port, and has been used by programs such as the Red Worm.

In this case, as the source port is typically 1214 for the hosts, is could possibly be the file sharing program Kazaa, or it could be a Red Worm infestation on MY.NET.150.133

Defensive Recommendations:

Watch traffic to and from this host carefully, and determine if it is Kazaa or a more serious problem. If Kazaa is not allowed, then block the ports is uses.

Correlations:

http://www.giac.org/practical/Robert_Turner_GCIA.doc

Back Orifice

This is the famous Trojan Back Orifice that allows remote control of a PC. This alert comes from a source UPD port of 31337. It appears to have been scanned for on numerous times, but it is difficult to tell without more information if there is an actual compromised host.

MISC source port 53 to <1024

This alert is for traffic from source port 53 (DNS) to a reserved port (<1024). This is typically a DNS request (Microsoft DNS in particular will do this). In these alerts, the traffic is from port 53 to port 53, so this is likely DNS.

Port 55850 tcp - Possible myserver activity - ref. 010313-1

To quote Mike Worman from the Incidents mailing list:

> "MyServer is a little known DDOS agent that was running around late in the summer. It binds to UDP 55850 and possibly TCP 55850, and the rootkit installs trojans of ls and ps, so you won't see it running. You WILL see it with netstat though."

http://archives.neohapsis.com/archives/incidents/2000-10/0136.html

TCP SRC and DST outside network

See section on UDP SRC and DST outside network. The main difference is that this is TCP instead of UDP.

INFO - ICQ Access

ICQ is a popular Instant Messaging program. This can be dangerous, as there was a recent buffer overflow announced for the ICQ client.

WEB-MISC whisker head

Whisker is a program that scans for vulnerable web server scripts by using a 'HEAD' rather than a GET. Using HEAD allowed it to fool some older IDS systems.

Incomplete Packet Fragments Discarded

A TCP stack will discard packet fragments if they haven't all arrived in a certain interval of time. This is the ICMP message that would be sent back to the client sending the packets.

EXPLOIT x86 NOOP

Many buffer overflows will use a long string of NOOP (no operation) OP Codes in their exploit code. This is because the attacker is not exactly sure of the memory alignment, so the JUMP will just hit this block of NOOPs and fall through to the actual exploit code.

ICMP traceroute

ICMP traceroute is typically used by Windows systems (Unix will often use a UPD traceroute). This is an attempt to map out the network between two hosts.

WEB-MISC 403 Forbidden

This is a 403 result code from a HTTP request. This means that the client requested a page he was not allowed to view.

WEB-MISC compaq nsight directory traversal

If a server is running the Compaq Nsight management tools, there is the possibility of a serious directory traversal problem that would allow the attacker out of the web root. This could allow view restricted files via a URL with ../. This is similar to the Unicode directory traversal problem.

References:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0771

Possible trojan server activity

To quote http://www.giac.org/practical/Garreth_jeremiah_GCIA.zip

> "This is a generic alert attempting to discover network traffic that may signify the use of trojan horse programs. The traffic causing this particular alert is either sourced or destined to port 27374 which is often attributed to the SubSeven Trojan (>= V2.2).
>
> The alert that generated this information is one created by the university:

alert tcp any any <> any 27374 (msg:"Possible trojan server activity";)"

SCAN Synscan Portscan ID 19104

This is synscan v1.8, which scans large blocks of IP addresses for specified services using a TCP ID of 19104. The TCP ID is what triggered Snort to alert on this.

References:
http://www.incidents.org/archives/intrusions/msg00696.html

INFO FTP anonymous FTP

This is someone logging in (or attempting to log in) to a FTP server using the account 'anonymous'.

INFO Outbound GNUTella Connect accept

This is the file sharing program GNUTella making outbound connections.

ICMP Echo Request Windows

A Windows TCP stack pinging someone (Echo Request).

NMAP TCP ping!

This is the scanning and fingerprinting tool NMAP.

MISC traceroute

Network mapping with unidentified traceroute clients.

WEB-CGI scriptalias access

Quoted From http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=2300

> NSCA httpd prior to and including 1.5 and Apache Web Server prior to 1.0 contain a bug in the ScriptAlias function that allows remote users to view the source of CGI programs on the web server, if a ScriptAlias directory is defined under DocumentRoot. A full listing of the CGI-BIN directory can be obtained if indexing is turned on, as well. This is accomplished by adding multiple forward slashes in the URL (see exploit). The web server fails to recognize that a ScriptAlias directory is actually redirected to a CGI directory when this syntax is used, and returns the text of the script instead of properly executing it. This may

allow an attacker to audit scripts for vulnerabilities, retrieve proprietary information, etc.

WEB-MISC Attempt to execute cmd

This is an attempt to run 'cmd.exe' within an HTTP request. These are typically seen as part of Unicode attacks on IIS.

INFO Possible IRC Access

This is Internet Relay Chat. Many Trojan horses connect back to IRC channels to establish connections out from behind firewalls.

SCAN Proxy attempt

Someone is scanning on the typical HTTP and SOCKS proxy port – 1080. If one is found, hackers can sometimes use these to hide their web surfing source.

EXPLOIT NTPDX buffer overflow

This is a buffer overflow attempt on the NTP (network time protocol).

References:
http://www.securityfocus.com/advisories/3200

ICMP Destination Unreachable (Communication Administratively Prohibited)

Someone appears to be pinging through a gateway that has an access control list to prevent the traffic. The gateway is returning this ICMP error message.

WEB-FRONTPAGE _vti_rpc access

This is access to IIS Front Page extensions. If they are not in use, this is probably an attempt at exploiting Front Page vulnerabilities.

WEB-IIS _vti_inf access

This is access to IIS Front Page extensions. If they are not in use, this is probably an attempt at exploiting Front Page vulnerabilities.

ICMP Echo Request BSDtype

This is a ping (echo request) from a BSD TCP stack.

Null scan!

This is a TCP packet with no options set, a sequence number of 0, and an ACK of 0. This can be used to scan past some simple packet filter firewalls.

References:
http://www.giac.org/practical/Garreth_jeremiah_GCIA.zip

255PARTY - Possible My Party infection

This is a Virus that spreads via mass mailing (Outlook)

References:
http://www.antivirus.com/vinfo/virusencyclo/default2.asp?m=q&virus=MYPARTY

FTP DoS ftpd globbing

Globbing is the use of wildcards in an expression. This is probably watching for a particular expression that can crash the FTP server, leading to a Denial of Service.

WEB-IIS view source via translate header

There was an old bug in IIS that allowed viewing of ASP source code by specifying 'translate f' in the header. This will sometimes false positive from Front Page editors.

ICMP Echo Request Nmap or HPING2

Crafted Echo Requests from either Nmap or HPING2. Both programs craft packets.

ICMP Router Selection

This is used with router discovery to broadcast for routers.

References:
http://www.iana.org/assignments/icmp-parameters

Watchlist 000220 IL-ISDN255-990517

This is a particular network they are watch for all traffic from. This network has probably caused them problems in the past.

spp_http_decode: CGI Null Byte attack detected

This is searching for the Unicode character %00 in the HTTP request. This is part of many web attacks, but is also part of normal HTTP traffic.

Reference:
http://archives.neohapsis.com/archives/snort/2000-11/0244.html

High port 65535 udp - possible Red Worm – traffic

This is a local rule showing traffic to or from the highest UPD port – 65535. They seem to think it can signify traffic from Red Worm. The TCP version of this traffic appeared to be associated with Kazaa, but this traffic has mostly 65535 on both ends. This could actually be Red Worm traffic.

INFO MSN IM Chat data

This is Microsoft Instant Message chat.

ICMP Echo Request L3retriever Ping

This is a ping from the program L3retriever.

SMB Name Wildcard

This is the Net Bios wildcard – used to enumerate shares/services from a Windows server. This is someone trying to get a list of all of the services available on the server.

References:
http://www.sans.org/y2k/052300-0800.htm
http://www.giac.org/practical/Garreth_jeremiah_GCIA.zip

MISC Large UDP Packet

As there is no error detection built into UDP, the packets are typically kept small to prevent problems. In my experience, this type of traffic is typically associated with NFS v2, which used UDP as a transport.

SNMP public access.

This is access to the public (read-only) SNMP. A network management station (like OpenView) will send many of these. Also, there is a lot of information that most servers' and network equipment will provide via public SNMP, so hackers like to use this in reconnaissance.

spp_http_decode: IIS Unicode attack detected

This is the famous Unicode attack, where the attacker can escape the web root by specifying special characters in his Unicode equivalent. Once out of the web root, the attacker can run commands as the IIS_USER account.

connect to 515 from inside

This is a connection to the printer port. There have been a number of buffer overflow problems with this service lately, so anyone not actually printing to the server should be considered an attacker.

Port Scan Info

This section is broken into the 5 days separately, as I did not have enough memory to run this as one large file. Even with 1 Gig of RAM, SnortSnarf used almost all of the available system resources for each day's data.

Day 1 (Jan 25):

Signature (click for sig info)	# Alerts	# Sources	# Dests
TCP **U*PRS* scan	1	1	1
TCP *2UA**SF scan	1	1	1
TCP ***A**SF scan	1	1	1
TCP *2*A**S* scan	1	1	1
TCP *2*A*R*F scan	1	1	1
TCP *****R*F scan	1	1	1
TCP 1*U*P*S* scan	1	1	1
TCP 1*U*P*** scan	1	1	1
TCP *2UAP**F scan	4	4	1
TCP *****RS* scan	6	1	1
TCP ***AP*S* scan	6	1	1
TCP ******** scan	82	15	3
TCP ****P*** scan	83	36	3
TCP ******S* scan	35765	294	5308
UDP scan	190710	332	1699

By far, the greatest number of scans were SYN scans (35765) and UDP scans to various ports (190710). The SYN scans are likely either someone attempting to be stealthy with his scan, failed connections, or a firewall or router black-holing the connections.

Within the UDP scan data, the host MY.NET.60.43 was the top talker, with 81,914 alerts. This host appears to be a harmless NTP (network time protocol server), which is causing these alerts.

The next highest alert host, MY.NET.153.171, appears to come under constant SYN scan. At closer look, however, this host appears to be a firewall or proxy server that is handling outbound web connections. Most of the SYN scans coming going to this host are internet sites on port 80, which may just mean that the destination host was unresponsive.

Day 2 (Jan 26):

Signature (click for sig info)	# Alerts	# Sources	# Dests
TCP 12****S* scan	4	1	1
TCP ******** scan	10	10	1
TCP ****P*** scan	50	35	2
TCP ******S* scan	21782	198	2867
UDP scan	144071	221	322

The breakdown is similar to the previous day. By far, the greatest number of scans were SYN scans (21782) and UDP scans to various ports (144071). The SYN scans are likely either someone attempting to be stealthy with their scan, failed connections, or a firewall or router black-holing the connections.

The host with the most UDP scans on this day was again MY.NET.60.43, which appears to the NTP server.

The host with the second most UPD scans was MY.NET.6.45, which was connecting form port 7000 to port 7001 on various internal machines. This appears to be an AFS file server. AFS is a distributed file system similar to NFS.

Day 3 (Jan 27):

Signature (click for sig info)	# Alerts	# Sources	# Dests
TCP *2U*PR*F scan	1	1	1

TCP *2UAPRS* scan	1	1	1
TCP *2U*PRSF scan	2	1	1
TCP ******** scan	6	4	1
TCP ****P*** scan	82	44	2
TCP ******S* scan	27159	198	3548
UDP scan	167042	224	318

The breakdown is similar to the previous day. By far, the greatest number of scans were SYN scans (27159) and UDP scans to various ports (167042). The SYN scans are likely either someone attempting to be stealthy with their scan, failed connections, or a firewall or router black-holing the connections.

The top talker was again the NTP server. Second on the list is MY.NET.6.50, and third was MY.NET.6.52.

Day 4 (Jan 28):

Signature (click for sig info)	# Alerts	# Sources	# Dests
TCP **UA**S* scan	1	1	1
TCP 1*U***S* scan	1	1	1
TCP 1*U**R** scan	1	1	1
TCP ***A*R*F scan	6	1	1
TCP 1****R** scan	10	6	3
TCP ****P*S* scan	54	53	1
TCP ****P*** scan	98	59	3
TCP ******** scan	125	13	4
TCP ******S* scan	95371	350	9985
UDP scan	499054	398	4062

The breakdown is similar to the previous day. By far, the greatest number of scans were SYN scans (27159) and UDP scans to various ports (167042). The SYN scans are likely either someone attempting to be stealthy with their scan, failed connections, or a firewall or router black-holing the connections.

The top talker was again what appears to be an NTP server. After this, MY.NET.108.141 comes in, which appears to have been extensively scanned from the outside (especially from 64.152.108.141.

Day 5 (Jan 29):

Signature (click for sig info)	# Alerts	# Sources	# Dests
TCP **U****F scan	1	1	1
TCP 12*****F scan	1	1	1
TCP 1***PRS* scan	1	1	1
TCP 1*UAPR** scan	1	1	1
TCP 12U*P**F scan	1	1	1
TCP *2**PR** scan	1	1	1
TCP *2UAPR** scan	1	1	1
TCP *2*A**** scan	1	1	1
TCP **U*P**F scan	1	1	1
TCP 12*A***F scan	1	1	1
TCP 12**P*SF scan	1	1	1
TCP 12**P**F scan	1	1	1
TCP **U**R** scan	1	1	1
TCP 12****** scan	1	1	1
TCP **U**RS* scan	1	1	1
TCP 12UA**S* scan	1	1	1
TCP 12**PR*F scan	1	1	1
TCP 12UA**** scan	1	1	1
TCP 1*U****F scan	1	1	1
TCP *****RSF scan	1	1	1
TCP *2*APR** scan	1	1	1
TCP **UA*RSF scan	1	1	1
TCP **UAPR*F scan	1	1	1

TCP 12*AP*SF scan	1	1	1
TCP *2U****F scan	1	1	1
TCP 12*APRSF scan	1	1	1
TCP 1*UA**SF scan	1	1	1
TCP 1*UA*R*F scan	1	1	1
TCP ***APR*F scan	1	1	1
TCP *2***R*F scan	1	1	1
TCP *2*A*RSF scan	1	1	1
TCP **U*P*S* scan	1	1	1
TCP 12**P*** scan	1	1	1
TCP **U*PRS* scan	1	1	1
TCP **U*PR** scan	1	1	1
TCP 12UAP*** scan	1	1	1
TCP 12*AP*** scan	1	1	1
TCP 1*UAP**F scan	1	1	1
TCP *2U**RS* scan	1	1	1
TCP 12U***SF scan	1	1	1
TCP *2**PR*F scan	1	1	1
TCP *2UA*R** scan	2	2	1
TCP *2U*P*** scan	2	2	2
TCP **U*P*SF scan	2	1	1
TCP 12****S* scan	2	2	2
TCP 12*A*R** scan	2	2	1
TCP *2U**R** scan	2	2	1
TCP ***A*R*F scan	3	3	2
TCP ***A*RS* scan	4	3	2
TCP 1****R** scan	20	6	2
TCP ******** scan	115	53	5

TCP ****P*** scan	118	74	4
TCP ******S* scan	79129	371	8727
UDP scan	432365	344	393

This day saw a large number of strange scans, although most were only seen once, and most were from different IP addresses. There is the possibility here of a distributed scan on this day.

Out of Spec Data:

Following is all of the Out of Spec data for the five day period:

```
01/25-14:24:25.974971 65.93.180.243:3476 -> MY.NET.88.162:1214
TCP TTL:110 TOS:0x0 ID:48811  DF
**SF**A* Seq: 0xB3CBCCE1   Ack: 0x5F5698EF   Win: 0x13F6
00 00 00 00 00 00                                ......
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

01/26-12:28:28.903591 145.236.140.74:1214 -> MY.NET.150.133:1214
TCP TTL:49 TOS:0x0 ID:21505  DF
21S***** Seq: 0xEE04156C   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 2572653 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/26-12:31:26.534501 145.236.140.74:1251 -> MY.NET.150.133:1214
TCP TTL:49 TOS:0x0 ID:62090  DF
21S***** Seq: 0xF82E745A   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 2590416 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/26-12:39:41.355563 145.236.140.74:1361 -> MY.NET.150.133:1214
TCP TTL:49 TOS:0x0 ID:32438  DF
21S***** Seq: 0x189636AC   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 2639898 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/26-12:43:38.677969 145.236.140.74:1417 -> MY.NET.150.133:1214
TCP TTL:49 TOS:0x0 ID:63693  DF
21S***** Seq: 0x27AA5C02   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 2663631 0 EOL EOL EOL EOL

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

01/27-06:25:02.953126 65.129.33.127:18245 -> MY.NET.5.96:21536
TCP TTL:21 TOS:0x0 ID:18458  DF
2*SFRP*U Seq: 0x2F62696E   Ack: 0x2F636F6D   Win: 0x6E2F
2E 70 6C 20 48 54 54 50 2F 31                    .pl HTTP/1

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/27-06:25:06.806710 65.129.33.127:18245 -> MY.NET.5.96:21536
```

```
TCP TTL:21 TOS:0x0 ID:20506  DF
2*SFRP*U Seq: 0x2F62696E   Ack: 0x2F636F6D   Win: 0x6E2F
2E 70 6C 3F 62 62 61 74 74 3D                    .pl?bbatt=

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

01/28-20:46:33.703718 64.166.209.137 -> MY.NET.88.162
TCP TTL:110 TOS:0x0 ID:33339  DF MF
Frag Offset: 0x0   Frag Size: 0x22
5A 1D 6B 5E 5B 1D 6C 99 22 37 5C 74 DD D3 2A 0C  Z.k^[.l."7\t..*.
C6 7A 15 8E E0 DC 01 2D 3E D6 87 7A D4 83 DF 32  .z.....->..z...2
3D B0                                            =.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/28-20:46:33.815778 64.166.209.137 -> MY.NET.88.162
TCP TTL:110 TOS:0x0 ID:33595  DF MF
Frag Offset: 0x0   Frag Size: 0x22
5B DC 9B FC 60 DE C0 BA E9 C4 23 F9 DA E5 95 D9  [...`.....#.....
E5 9A C7 D1 02 A6 EA 8D E4 6F 39 A3 53 B2 EB 18  .........o9.S...
75 57                                            uW

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

The first 5 of these alerts are associated with port 1214, which is possibly Kazaa – a file swapping program. Given that these are peer-to-peer connections to unknown machines, seeing a few broken packets probably isn't a large concern. If this isn't Kazaa, then this could be a sign of a covert channel.

The next two,

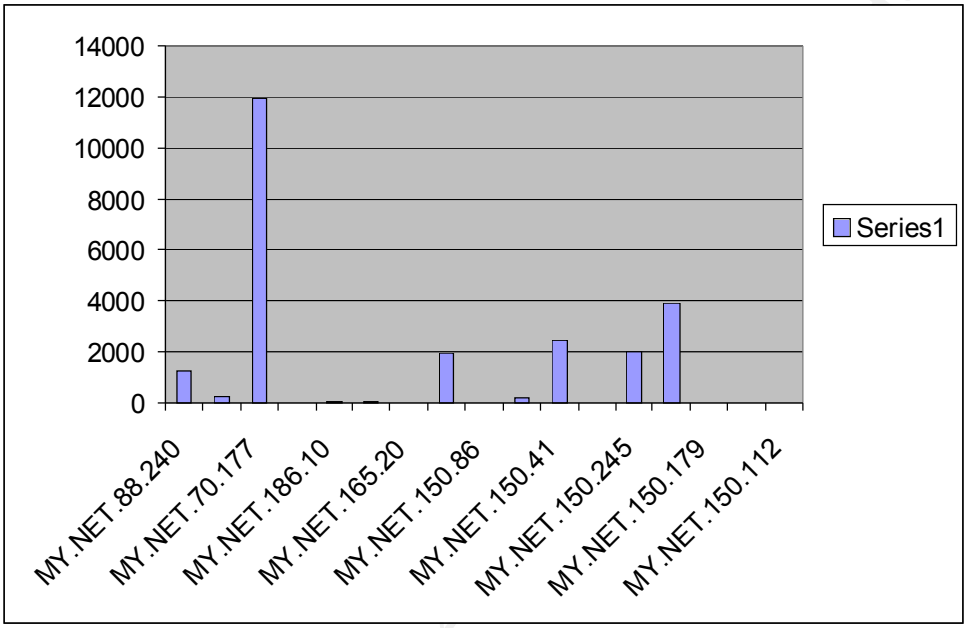
```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

01/27-06:25:02.953126 65.129.33.127:18245 -> MY.NET.5.96:21536
TCP TTL:21 TOS:0x0 ID:18458  DF
2*SFRP*U Seq: 0x2F62696E   Ack: 0x2F636F6D   Win: 0x6E2F
2E 70 6C 20 48 54 54 50 2F 31                    .pl HTTP/1

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
01/27-06:25:06.806710 65.129.33.127:18245 -> MY.NET.5.96:21536
TCP TTL:21 TOS:0x0 ID:20506  DF
2*SFRP*U Seq: 0x2F62696E   Ack: 0x2F636F6D   Win: 0x6E2F
2E 70 6C 3F 62 62 61 74 74 3D                    .pl?bbatt=

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

are strange. Not only are the TCP flags set illegally, but also this appears to be HTTP traffic destined to MY.NET.5.96 on port 96. This is worth looking into.

The last two alerts are packet fragments.

Link graph of selected data:

Following is a graph on the sources of the SNMP Public alerts. Since the sources were all internal, I wanted to determine if there were a few machines making most of the alerts, or if the alerts were more uniform and distributed. As can be seen, one host really stands out (MY.NET.70.177), which is possibly a network management station such as HP Openview. Five other hosts also had a large number of alerts attributed to them – possibly being other network management stations, Jet Direct administration terminals, or even compromised hosts that were being used to scan the rest of the internal network. As none of the hosts appeared to be connecting to a large number of diverse IP addresses, they probably were not being used for reconnaissance scanning.



Bibliography:

Bell, Mike. GCIA Practical for Capitol SANS/Washington DC. December 2000 – January 2001. URL: http://www.sans.org/y2k/practical/Mike_Bell_GCIA.doc (3 Feb 2002).

Berners-Lee, T. Uniform Resource Identifiers (URI): Generic Syntax. 19 Aug 2002. URL: http://www.ietf.org/rfc/rfc2396.txt (3 Feb 2002).

Brian (bmc@snort.org). Re: [Snort-users] uricontent misbehaving? 2 Nov 2001. URL: http://archives.neohapsis.com/archives/snort/2001-11/0177.html (2 Feb 2002).

Dejtlich, Richard. Interpreting Network Traffic: A Network Intrusion Detector's Look at Suspicious Events. v 2.8. 14 May 00. URL: http://home.satx.rr.com/bejtlich/intv2-8.html (1 Feb 2002).

Dixon, Jason. RE [Incidents] New DNS connection with SYN ACK. 14 Jan 2002. URL: http://www.securityfocus.com/archive/75/250180 (Feb 2 2002).

EEYE Digital Security – description of CGI scans. URL: http://www.eeye.com/html/Support/Retina/RTHs/CGI_Scripts/572.html (3 Feb 2002).

Erdelyi, Gergely. F-Secure Virus Descriptions. Jul 2001. URL: http://www.europe.f-secure.com/v-descs/bady.shtml (29 Jan 2002).

Jeremiah, Garreth. Garreth Jeremiah's Submission for GCIA Intrusion Detection In-depth,Version 2.9. SANS 2001, Baltimore. URL: http://www.giac.org/practical/Garreth_jeremiah_GCIA.zip (Feb 2 2002).

Martin, Daniel. RE [Incidents] fast ssh scans. 18 Oct 2001. URL: http://www.securityfocus.com/archive/75/221281 (Jan 30 2002).

Microsoft Corp. Microsoft Internet Information Server 4.0 Security Checklist. Last Updated 27 Sep 1999. URL: http://www.fulgan.com/delphi/CheckList.htm (Jan 31 2002).

Mitre.Org. Common Vulnerabilities and Exposures. CAN-1999-0454 (under review). 28 Jul 1999. URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0454 (Feb 3 2002).

Mitre.Org Database. Common Vulnerabilities and Exposures. CVE-1999-0172. 11 Sep 1999. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0172 (Feb 2 2002).

Mitre.Org Database. Common Vulnerabilities and Exposures. CVE-1999-0771. 4 Jan 2000. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0771 (Feb 1 2002).

The NetBSD Foundation. NetBSD Security Advisory 2001-004. 5 Apr 2001. URL: http://www.securityfocus.com/advisories/3200 (Feb 3 2002).

Northcutt, Stephen. Global Incident Analysis Center – Detects Analyzed 5/23/00. 23 May 2000. URL: http://www.sans.org/y2k/052300-0800.htm (Feb 2 2002).

Plant, Simon. A Breakdown of the Top Five Windows 2000 IIS Threats in 2001. 18 Aug 2001. URL: http://rr.sans.org/win2000/5threats.php (Jan 31 2002).

Postel, Jon et al. ICMP TYPE NUMBERS. Last Updated 27 Aug 2001. URL: http://www.iana.org/assignments/icmp-parameters (Jan 30 2002).

Puppy, Rainforest. Whisker information, scripts, and updates. Last updated 26 Jan 2002. URL: http://www.wiretrip.net/rfp/p/doc.asp/i7/d21.htm (3 Feb. 2002).

Roesch, Martin. How to Write Snort Rules and Keep Your Sanity. Snort Users Manual Release 1.8.3. URL: http://www.snort.org/docs/writing_rules/chap2.html#tth_sEc2.3.30 (3 Feb 2002).

Security Focus Vulnerability Database. FormMail Recipient CGI Variable Spamming Vulnerability. Updated 25 Jan 2002. URL: http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=2469 (28 Jan 2002).

Security Focus Vulnerability Database. NCSA/Apache httpd ScriptAlias Source Retrieval Vulnerability. 25 Sep 1999. URL: http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=2300 (Feb 2 2002).

Security Team List of Exploits. Additional details about the IIS remote execution vulnerability. 27 Oct 2000. URL: http://www.securiteam.com/exploits/Additional_details_about_the_IIS_remote_execution_vulnerability.html (Feb 2 2002)

Simard, Louis-Eric. RE: [Incidents] New game using port 1470? 7 May 2000. URL: http://www.securityfocus.com/archive/75/59075 (Jan 30 2002).

Siske, Andrew. GIAC Intrusion Detection Practical Assignment for SANS Security DC 2000. July 5 – 10, 2000. URL: http://www.sans.org/y2k/practical/Andy_Siske_GCIA.htm (Jan 29 2002).

Smith, Donald. RE [Intrusions] This had me stumped… 8 Jun 2001. URL: http://www.incidents.org/archives/intrusions/msg00696.html (Feb 2 2002).

Snort Signature Database. Last Update 11 Jan 2002. URL: http://www.snort.org/snort-db/sid.html?id=1010 (Feb 3 2002).

Stewart, Joe. RE [Snort-users] CGI Null Byte Attack. 20 Nov 2000. URL: http://archives.neohapsis.com/archives/snort/2000-11/0244.html (Jan 30 2002).

Trend Micro Virus Encyclopedia. URL: http://www.antivirus.com/vinfo/virusencyclo/default2.asp?m=q&virus=MYPARTY (Feb 2 2002).

Turner, Robert. GIAC Level Two: Intrusion Detection In Depth -- SANS Parliament Square, London June 20-23, 2001. URL: http://www.giac.org/practical/Robert_Turner_GCIA.doc (Feb 3 2002).

Worman, Mike. RE [Incidents] Connections from unknown. 23 Oct 2000. URL: http://archives.neohapsis.com/archives/incidents/2000-10/0136.html (Jan 30 2002).

Ziehe, Soeren. [Incidents] Formmail. 01 Sept. 2001. URL: http://archives.neohapsis.com/archives/incidents/2001-08/0446.html (Jan 31 2002).