Kevin Timm

GCIA Version 3.1

SANS Lonestar Conference
San Antonio, Texas
March 10 – 16 2002

Assignment 1: State Of Intrusion Detection

IDS Evasion with Session Splicing

IDS vendors have made tremendous gains in defeating IDS evasion techniques within the past two years, especially where string matching and string obfuscation is concerned. Evasion techniques that are network based are not as easy for an IDS to defend against as string obfuscation techniques. These network level evasion techniques were first brought to the forefront in the landmark 1998 research paper "Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection" by Thomas Ptacek and Timothy Newsham[1]. The paper details specific network level problems specifically with fragmentation and session re-assembly. Recently tools such as Whisker by RainForestPuppy[2] and Nessus[3] the vulnerability scanner have implemented session splicing techniques. Many techniques outlined by Ptacek and Newsham are common to both fragmentation and session splicing. Session splicing, because it involves a session is useful only when the payload can be delivered over multiple packets where fragmentation can be used with all protocols. This paper will focus on IDS evasion using session splicing timeouts and to a lesser degree rule alerting and first exit problems that session splicing can create.

The basic premise behind session splicing is to deliver the payload over multiple packets thus defeating simple pattern matching without session reconstruction. This payload can be delivered in many different manners and even spread out over a long period of time. Currently, Whisker and Nessus have session splicing capabilities, and other tools exist in the wild (see detect #1 from the detects section). I have created a perl script splicer.pl that I will use to demonstrate many of these techniques. A trademark of a spliced session is a continuous stream of small packets. The network trace below shows a standard spliced session. For readability, this request has been abbreviated removing bytes nine thru 31.

Trademark Session Splice

```
17:06:22.252252 64.194.107.85.32787 > W.X.Y.106.80: S 848344882:848344882(0) win 5840 <mss
1460,sackOK,timestamp 3152623[|tcp]> (DF) (ttl 64, id 44509, len 60)
17:06:22.292252 W.X.Y.106.80 > 64.194.107.85.32787: S 268545229:268545229(0) ack 848344883 win
5792 <mss 1460,sackOK,timestamp 17985824[|tcp]> (DF) (ttl 53, id 0, len 60)
17:06:22.292252 64.194.107.85.32787 > W.X.Y.106.80: . [tcp sum ok] ack 1 win 5840
<nop,nop,timestamp 3152627 17985824> (DF) (ttl 64, id 44510, len 52)
17:06:22.292252 64.194.107.85.32787 > W.X.Y.106.80: P [tcp sum ok] 1:2(1) ack 1 win 5840
<nop,nop,timestamp 3152627 17985824> (DF) (ttl 64, id 44511, len 53)
17:06:22.342252 W.X.Y.106.80 > 64.194.107.85.32787: . [tcp sum ok] ack 2 win 5792
<nop,nop,timestamp 17985829 3152627> (DF) (ttl 53, id 56810, len 52)
17:06:22.492252 64.194.107.85.32787 > W.X.Y.106.80: P [tcp sum ok] 2:3(1) ack 1 win 5840
<nop,nop,timestamp 3152647 17985829> (DF) (ttl 64, id 44512, len 53)
17:06:22.532252 W.X.Y.106.80 > 64.194.107.85.32787: . [tcp sum ok] ack 3 win 5792
<nop,nop,timestamp 17985848 3152647> (DF) (ttl 53, id 56811, len 52)
17:06:22.692252 64.194.107.85.32787 > W.X.Y.106.80: P [tcp sum ok] 3:4(1) ack 1 win 5840
<nop,nop,timestamp 3152667 17985848> (DF) (ttl 64, id 445F13, len 53)
```

```
17:06:22.732252 W.X.Y.106.80 > 64.194.107.85.32787: . [tcp sum ok] ack 4 win 5792
<nop,nop,timestamp 17985868 3152667> (DF) (ttl 53, id 56812, len 52)
17:06:22.892252 64.194.107.85.32787 > W.X.Y.106.80: P [tcp sum ok] 4:5(1) ack 1 win 5840
<nop,nop,timestamp 3152687 17985868> (DF) (ttl 64, id 44514, len 53)
17:06:22.932252 W.X.Y.106.80 > 64.194.107.85.32787: . [tcp sum ok] ack 5 win 5792
<nop,nop,timestamp 17985888 3152687> (DF) (ttl 53, id 56813, len 52)
17:06:23.092252 64.194.107.85.32787 > W.X.Y.106.80: P [tcp sum ok] 5:6(1) ack 1 win 5840
<nop,nop,timestamp 3152707 17985888> (DF) (ttl 64, id 44515, len 53)
17:06:23.122252 W.X.Y.106.80 > 64.194.107.85.32787: . [tcp sum ok] ack 6 win 5792
<nop,nop,timestamp 17985908 3152707> (DF) (ttl 53, id 56814, len 52)
17:06:23.292252 64.194.107.85.32787 > W.X.Y.106.80: P [tcp sum ok] 6:7(1) ack 1 win 5840
<nop,nop,timestamp 3152727 17985908> (DF) (ttl 64, id 44516, len 53)
17:06:23.332252 W.X.Y.106.80 > 64.194.107.85.32787: . [tcp sum ok] ack 7 win 5792
<nop,nop,timestamp 17985928 3152727> (DF) (ttl 53, id 56815, len 52)
17:06:23.492252 64.194.107.85.32787 > W.X.Y.106.80: P [tcp sum ok] 7:8(1) ack 1 win 5840
<nop,nop,timestamp 3152747 17985928> (DF) (ttl 64, id 44517, len 53)
17:06:23.532252 W.X.Y.106.80 > 64.194.107.85.32787: . [tcp sum ok] ack 8 win 5792
<nop,nop,timestamp 17985948 3152747> (DF) (ttl 53, id 56816, len 52)
17:06:28.292252 64.194.107.85.32787 > W.X.Y.106.80: P [tcp sum ok] 31:32(1) ack 1 win 5840
<nop,nop,timestamp 3153227 17986408> (DF) (ttl 64, id 44541, len 53)
17:06:28.322252 W.X.Y.106.80 > 64.194.107.85.32787: . [tcp sum ok] ack 32 win 5792
<nop,nop,timestamp 17986428 3153227> (DF) (ttl 53, id 56840, len 52)
17:06:28.492252 64.194.107.85.32787 > W.X.Y.106.80: P [tcp sum ok] 32:33(1) ack 1 win 5840
<nop,nop,timestamp 3153247 17986428> (DF) (ttl 64, id 44542, len 53)
17:06:28.532252 W.X.Y.106.80 > 64.194.107.85.32787: . [tcp sum ok] ack 33 win 5792
<nop,nop,timestamp 17986448 3153247> (DF) (ttl 53, id 56841, len 52)
17:06:28.692252 64.194.107.85.32787 > W.X.Y.106.80: P 33:83(50) ack 1 win 5840 <nop,nop,timestamp
3153267 17986448> (DF) (ttl 64, id 44543, len 102)
17:06:28.732252 W.X.Y.106.80 > 64.194.107.85.32787: . [tcp sum ok] ack 83 win 5792
<nop,nop,timestamp 17986468 3153267> (DF) (ttl 53, id 56842, len 52)
17:06:28.732252 W.X.Y.106.80 > 64.194.107.85.32787: F [tcp sum ok] 566:566(0) ack 83 win 5792
<nop,nop,timestamp 17986468 3153267> (DF) (ttl 53, id 56844, len 52)
17:06:28.732252 64.194.107.85.32787 > W.X.Y.106.80: . ack 1 win 5840 <nop,nop,timestamp 3153271
17986468,nop,nop,[|tcp]> (DF) (ttl 64, id 44544, len 64)
17:06:28.742252 W.X.Y.106.80 > 64.194.107.85.32787: P 1:566(565) ack 83 win 5792
<nop,nop,timestamp 17986468 3153267> (DF) (ttl 53, id 56843, len 617)
17:06:28.742252 64.194.107.85.32787 > W.X.Y.106.80: . [tcp sum ok] ack 567 win 6780
<nop,nop,timestamp 3153272 17986468> (DF) (ttl 64, id 44545, len 52)
17:06:28.742252 64.194.107.85.32787 > W.X.Y.106.80: F [tcp sum ok] 83:83(0) ack 567 win 6780
<nop,nop,timestamp 3153272 17986468> (DF) (ttl 64, id 44546, len 52)
17:06:28.792252 W.X.Y.106.80 > 64.194.107.85.32787: . [tcp sum ok] ack 84 win 5792
<nop,nop,timestamp 17986474 3153272> (DF) (ttl 53, id 56845, len 52)
```

At first this may seem easy to defend against by just checking for abnormally small packets. However, this is not the case because these are packets with the ACK flag set and packets with the ACK flag set are normally very small and contain no payload. This makes detection under normal traffic more difficult.

For testing and comparison, a lab was constructed that contains three hosts on the same broadcast domain. The initial setup of the lab was as follows using Snort, Cisco Secure IDS and a Windows 2000 web server running Entercept 2.01 host IDS agent.

A full description of the Cisco IDS log format and signatures used in these examples can be found in Appendix B. It should be noted that all request returned a page of some sort from the web server.

Lab Setup

Host Type	IP	Version	Function
Cisco Secure IDS	W.X.Y.123	3.1 Beta	IDS Only
Snort Sensor	W.X.Y.122	Snort 1.8.6 / RH 7.2	IDS / Apache Web Server
Windows Host IDS	W.X.Y.124	W2K Server / Entercept 2.01	IIS Web Server HIDS Agent

Several different requests were sent using the splicer.pl tool. Tables depict the request, which is the web request, splice size in bytes and timing in seconds with the associated logs from the devices. These attacks were sent from the ip 64.194.107.85. Note: organization specific information has been X out in the Cisco IDS logs.

Detecting Session splicing with string matching IDS is very challenging. It is not possible to create signatures that just look for small packets with the ACK flag set since this occurs abundantly in normal traffic. Signatures must look do some payload comparison. Snort has several signatures to detect the tool Whisker session splices.

Snort Session Splicing Signature 1

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-MISC
whisker splice attack"; content: "|20|"; flags: A+; dsize: 1;reference:arachnids,296;
classtype:attempted-recon; sid:1104; rev:1;)
```

Snort Session Splicing Signature 2

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-MISC
whisker splice attack"; dsize: <5; flags: A+; content:
"|09|";reference:arachnids,415; classtype:attempted-recon; sid:1087; rev:1;)
```

The first signature looks for 0x20 (a space) in the first 2 bytes of a packet with ack flags set and directed toward the defined variable $HTTP_SERVERS. The second signature looks for 0x09 (tab) within the first six bytes of a packet with ACK flags set directed toward the defined variable $HTTP_SERVERS. These signatures catch the default Whisker session splicing usage usage fine. However, these rules can be evaded by splicing the session differently. The tool splicer.pl allows different values to be specified for the payload size to be delivered. It should be pointed out that splicer.pl is written for HTTP traffic, however the basic premise of session splicing extends into all TCP protocols.

Example 1: Non Malicious Spliced request

<table>
<tr><th>Request</th><th>Splice Size (bytes)</th><th>Timing (seconds)</th></tr>
<tr><td>Get /index.html HTTP/1.0\r\n</td><td>1</td><td>.20</td></tr>
<tr><th colspan="3">Cisco IDS Logs</th></tr>
<tr><td colspan="3">None Available</td></tr>
<tr><th colspan="3">Snort Logs</th></tr>
<tr><td colspan="3">04/09-22:07:34.680000 [**] [1:1104:1] WEB-MISC whisker splice attack [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 64.194.107.85:33185 -> W.X.Y.124:80
04/09-22:07:36.480000 [**] [1:1104:1] WEB-MISC whisker splice attack [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 64.194.107.85:33185 -> W.X.Y.124:80
04/09-22:07:36.560000 [**] [1:0:0] Session Splice WEB [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 64.194.107.85:33185 -> W.X.Y.124:80
04/09-22:07:37.370000 [**] [1:0:0] Session Splice WEB [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 64.194.107.85:33185 -> W.X.Y.124:80</td></tr>
</table>

What we can ascertain from the initial test is that the Cisco IDS has no default check for generic Whisker style (1 byte) session splicing if there is not a malicious payload associated with the request. Snort generated four alerts associated with two signatures, one of which is associated with Whisker and one signature that is a custom generic splicing signature that I developed, which will be detailed later.

To avoid the standard Whisker signatures an attacker can increase the size of the spliced packets as demonstrated below.

Session splicing with larger payloads

```
23:56:36.991823 64.194.107.85.33244 > W.X.Y.124.80: S 3136238607:3136238607(0) win 5840 <mss
1460,sackOK,timestamp 15635196[|tcp]> (DF)
23:56:37.091823 W.X.Y.124.80 > 64.194.107.85.33244: S 1360132749:1360132749(0) ack 3136238608
win 17520 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
23:56:37.091823 64.194.107.85.33244 > W.X.Y.124.80: . ack 1 win 5840 <nop,nop,timestamp 15635206
0> (DF)
23:56:37.091823 64.194.107.85.33244 > W.X.Y.124.80: P 1:3(2) ack 1 win 5840 <nop,nop,timestamp
15635206 0> (DF)
23:56:37.301823 W.X.Y.124.80 > 64.194.107.85.33244: . ack 3 win 17518 <nop,nop,timestamp 3198390
15635206> (DF)
23:56:38.091823 64.194.107.85.33244 > W.X.Y.124.80: P 3:5(2) ack 1 win 5840 <nop,nop,timestamp
15635306 3198390> (DF)
23:56:38.311823 W.X.Y.124.80 > 64.194.107.85.33244: . ack 5 win 17516 <nop,nop,timestamp 3198400
15635306> (DF)
23:56:39.091823 64.194.107.85.33244 > W.X.Y.124.80: P 5:7(2) ack 1 win 5840 <nop,nop,timestamp
15635406 3198400> (DF)
23:56:39.311823 W.X.Y.124.80 > 64.194.107.85.33244: . ack 7 win 17514 <nop,nop,timestamp 3198410
15635406> (DF)
23:56:40.091823 64.194.107.85.33244 > W.X.Y.124.80: P 7:9(2) ack 1 win 5840 <nop,nop,timestamp
15635506 3198410> (DF)
```

Example 2: Splicing with larger splices

Request	Splice Size	Timing
/index.html	2	1
Cisco IDS Logs		
None Available		
Snort Logs		
04/10-00:52:06.690000 [**] [1:0:0] Session Splice WEB [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 64.194.107.85:33243 -> W.X.Y.124:80 04/10-00:52:09.680000 [**] [1:0:0] Session Splice WEB [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 64.194.107.85:33243 -> W.X.Y.124:80		

This successfully defeats Snorts Whisker based evasion signatures. To write a signature to detect this evasion technique, it is required that the IDS look for smaller packets with the ACK flag set and some content. There must be some content associated with the packet otherwise the signature will trigger on all associated ACK packets. The generic splicing rule below will work.

Generic Session Splicing rule

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"Generic Session
Splice ATTACK"; uricontent: "H"; flags: A+; dsize: <16; classtype:bad-unknown;)
```

This particular signature detects any URI with a content of “H” and a dsize of less than 17 bytes. This should catch any web request that does not conform to “$METHOD / HTTP/1.0\r\n\” where $METHOD is equivalent to any HTTP method. The “H” was chosen because it is necessary in the HTTP request. This is still quite easy to evade by padding the URI with self referencing “./”directories, using a tab delimiter (Apache) or a space (IIS and Apache) and Nulls (IIS). This example will use the ida overflow URL since it does not require a specific URL for the attack to be successful. The Snort signature for the ida overflow is as follows

Snort ida overflow signature

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS ISAPI
.ida attempt"; uricontent:".ida?"; nocase; dsize:>239; flags:A+;
reference:arachnids,552; classtype:web-application-attack; reference:cve,CAN-
2000-0071; sid:1243; rev:2;)
```

This signature looks for the URI content of ida? with a size of 240 bytes. There are a couple methods that can evade this. One method is to splice the payload into requests that are larger, such as 18 bytes, thus evading the string match. Now this signature could be re-written to only look for ida? and not require a length of 240 bytes. This signature will then trigger on normal traffic and create false alarms. Since this attack does not require a specific URI to be requested, an attacker could front load the request with bogus data to create a longer URI. To make this work we need to calculate the amount of characters needed to have the "H" fall into an 18 byte slice. We needed four bytes at the front of our request so the request ends up looking like the request below.

Request for Example 3:

```
./splicer.pl -h W.X.Y.124 -r "/NNNN.ida?xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx" -t 1 -s 18
```

Example 3: Using padding to evade session signatures

Request	Splice Size	Timing
Above Example 3	1	18
Cisco IDS Logs		
4,1001256,2002/04/11,01:17:49,2002/04/10,20:17:49,10008,100,101,OUT,IN,5,5126, 0,TCP/IP,64.194.107.85,W.X.Y.124,32768,80,0.0.0.0,		
Snort Logs		
04/10-20:17:46.150000 [**] [1:1242:2] WEB-IIS ISAPI .ida access [**] [Classification: access to a potentually vulnerable web application] [Priority: 2] {TCP} 64.194.107.85:32768 -> W.X.Y.124:80		

Notice that Cisco Secure IDS successfully detects the true attack with a signature which depicts the true threat. Snort detects ida access, but does not trigger the attack signature that it should. Even though Snort alerts this alert appears less threatening than it really is.

Session Re-assembly needed

Session splicing signatures for string matching IDS devices are proven inadequate. Sessions must be re-assembled before comparison or else they will fail. Several different attacks were sent to the victim host for testing purposes.

In these examples, a malicious Unicode request was sent to the victim host. This signature was chosen because it's almost identical on Cisco and Snort. Cisco detects three signatures 3215, 5114 and 3216. These are all associated with general Unicode activity. Signature 5114 detects Unicode characters while signatures 3215 and 3216 detect the "/../" activity. Using 1 byte session splices, Snort only detected the previously mentioned session splicing signatures. This is an example of Snort's use of first exit rules.

Example 4: Malicious Spliced request

Request	Splice Size (bytes)	Timing (seconds)
GET /scripts/..%c0%af../ HTTP/1.0\r\n	1	.20
Cisco IDS Logs		
4,1008202,2002/04/10,03:57:55,2002/04/09,22:57:55,10008,100,101,OUT,IN,1,3000,80,TCP/IP,64.194.107.85,W.X.Y.124,33194,80,0.0.0.0,5634510 4,1008203,2002/04/10,03:58:02,2002/04/09,22:58:02,10008,100,101,OUT,IN,4,3215,0,TCP/IP,64.194.107.85,W.X.Y.124,33194,80,0.0.0.0,URL with /..,474554202F736372697074732F2E2EZZ 4,1008204,2002/04/10,03:58:06,2002/04/09,22:58:06,10008,100,101,OUT,IN,4,5114,1,TCP/IP,64.194.107.85,W.X.Y.124,33194,80,0.0.0.0,..%c0%af..*HTTP,474554202F736372697074732F2E2E2563302561662E2EZZ 4,1008205,2002/04/10,03:58:06,2002/04/09,22:58:06,10008,100,101,OUT,IN,5,3216,0,TCP/IP,64.194.107.85,W.X.Y.124,33194,80,0.0.0.0,../..,474554202F736372697074732F2E2E2563302561662E2EZZ		
Snort Logs		
04/09-22:57:56.090000 [**] [1:1104:1] WEB-MISC whisker splice attack [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 64.194.107.85:33194 -> W.X.Y.124:80 04/09-22:58:06.590000 [**] [1:1104:1] WEB-MISC whisker splice attack [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 64.194.107.85:33194 -> W.X.Y.124:80 04/09-22:58:07.090000 [**] [1:0:0] Session Splice WEB [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 64.194.107.85:33194 -> W.X.Y.124:80		

Using larger slices, we can evade the splicing signatures but are picked up by Snort's normal signatures. Cisco alerts to the true attack.

Example 5: Session Splicing using larger splices

Request	Splice Size	Timing
GET /scripts/..%c0%af../ HTTP/1.0\r\n	11	.5
Cisco IDS Logs		
,1001332,2002/04/11,02:12:37,2002/04/10,21:12:37,10008,100,101,OUT,IN,4,3215,0,TCP/IP,64.194.107.85,W.X.Y.124,32783,80,0.0.0.0,URL with /..,474554202F736372697074732F2E2E2563302561662EZZ 4,1001333,2002/04/11,02:12:37,2002/04/10,21:12:37,10008,100,101,OUT,IN,4,5114,1,TCP/IP,64.194.107.85,W.X.Y.124,32783,80,0.0.0.0,..%c0%af..*HTTP,474554202F736372697074732F2E2E2563302561662E2E2F20485454502F312E30ZZ 4,1001334,2002/04/11,02:12:37,2002/04/10,21:12:37,10008,100,101,OUT,IN,5,3216,0,TCP/IP,64.194.107.85,W.X.Y.124,32783,80,0.0.0.0,../..,474554202F736372697074732F2E2E2563302561662E2E2F20485454502F312E30ZZ		
Snort Logs		
04/10-21:12:38.140000 [**] [1:1113:1] WEB-MISC http directory traversal [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 64.194.107.85:32783 -> W.X.Y.124:80 04/10-21:12:38.640000 [**] [1:0:0] Abnormal WEB Request [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 64.194.107.85:32783 -> W.X.Y.124:80 04/10-21:12:39.360000 [**] [1:1287:2] WEB-IIS scripts access [**] [Classification: access to a potentually vulnerable web application] [Priority: 2] {TCP} 64.194.107.85:32783 -> W.X.Y.124:8		

The next attempt both IDS devices alert fine. This attack uses two byte splices and slightly delayed time sequences of one second. I think some assumptions can now be made. Both Snort and Cisco have some session re-assembly capabilities. Snort, however will alert differently for the same attack depending on techniques used. This has to do with how Snort determines which rule it will alert on.

Example 6: Smaller Splice time delayed padded HTTP

Request	Splice Size	Timing
GET /scripts/..%c0%af../ HTTP/1.0\r\n	2	1
Cisco IDS Logs		
4,1008380,2002/04/10,04:37:42,2002/04/09,23:37:42,10008,100,101,OUT,IN,4,3215,0,TCP/IP,64.194.107 85,W.X.Y.124,33234,80,0.0.0.0,URL with /..,474554202F736372697074732F2E2E25ZZ 4,1008381,2002/04/10,04:37:46,2002/04/09,23:37:46,10008,100,101,OUT,IN,4,5114,1,TCP/IP,64.194.107.85,W.X.Y.124,33234,80,0.0.0.0,..%c0%af..*HTTP,474554202F736372697074732F2E2E2563302561662E2E2FZZ 4,1008382,2002/04/10,04:37:46,2002/04/09,23:37:46,10008,100,101,OUT,IN,5,3216,0,TCP/IP,64.194.107.85,W.X.Y.124,33234,80,0.0.0.0,../..,474554202F736372697074732F2E2E2563302561662E2E2FZZ		
Snort Logs		
04/09-23:29:32.280000 [**] [1:1113:1] WEB-MISC http directory traversal [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 64.194.107.85:33233 -> W.X.Y.124:80 04/09-23:29:34.380000 [**] [110:4:1] spp_unidecode: Invalid Unicode String detected [**] {TCP} 64.194.107.85:33233 -> W.X.Y.124:80 04/09-23:37:51.590000 [**] [1:1287:2] WEB-IIS scripts access [**] [Classification: access to a potentually vulnerable web application] [Priority: 2] {TCP} 64.194.107.85:33234 -> W.X.Y.124:80		

The next example the HTTP portion of the request was padded to avoid triggering the generic splicing rule. The splices remain at 2 bytes. The time between splices was moved to 20 seconds. Snort appears to time out some of the request because this time a different alert is received. Snort must have been able to only re-assemble a portion of the attack.

Example 7: Smaller Slices, padded HTTP, longer time

Request	Splice Size	Timing
GET /scripts/..%c0%af../ HTTP/1.0\r\n	2	20
Cisco IDS Logs		
4,1008392,2002/04/10,04:47:40,2002/04/09,23:47:40,10008,100,101,OUT,IN,4,3215, 0,TCP/IP,64.194.107.85,W.X.Y.124,33236,80,0.0.0.0,URL with/..,474554202F736372697074732F2E2E25ZZ4,1008393,2002/04/10,04:49:00,2002/04/09,23:49:00, 10008,100,101,OUT,IN,4,5114,1,TCP/IP,64.194.107.85,W.X.Y.124,33236,80,0.0.0.0,..%c0%af..* HTTP,474554202F736372697074732F2E2E2563302561662E2E2FZZ4,1008394,2002/04/10,04:49:00,2002/04/09,23:49:00,10008,100,101,OUT,IN,5,3216,0,TCP/IP,64.194.107.85,W.X.Y.124,33236,80,0.0.0.0,../..,474554202F736372697074732F2E2E2563302561662E2E2FZZ		
Snort Logs		
04/09-23:50:59.380000 [**] [1:1287:2] WEB-IIS scripts access [**] [Classification: access to a potentually vulnerable web application] [Priority: 2] {TCP} 64.194.107.85:33236 -> W.X.Y.124:80		

In a default configuration, Snort appears to time out requests after a certain amount of time. Cisco is still seeing the alerts. The following request still uses two byte splices, but the time between splices is increased to 45 seconds.

Example 8: 2 Byte splices delayed by 45 seconds

<table>
<tr><th>Request</th><th>Splice Size</th><th>Timing</th></tr>
<tr><td>GET /scripts/..%c0%af../ HTTP/1.0\r\n</td><td>2</td><td>45</td></tr>
<tr><th colspan="3">Cisco IDS Logs</th></tr>
<tr><td colspan="3">4,1008411,2002/04/10,05:03:34,2002/04/10,00:03:34,10008,100,101,OUT,IN,4,3215,0,TCP/IP,64.194.107.85,W.X.Y.124,33239,80,0.0.0.0,URL with /..,4745474554202F73637263726970747 32F2E2E25ZZ
4,1008414,2002/04/10,05:06:34,2002/04/10,00:06:34,10008,100,101,OUT,IN,4,5114,1,TCP/IP,64.194.107.85,W.X.Y.124,33239,80,0.0.0.0,..%c0%af..*HTTP,4745474554202F7363726372697074732F2E2E2563302561662E2E2FZZ
4,1008415,2002/04/10,05:06:34,2002/04/10,00:06:34,10008,100,101,OUT,IN,5,3216,0,TCP/IP,64.194.107.85,W.X.Y.124,33239,80,0.0.0.0,../..,4745474554202F7363726372697074732F2E2E2563302561662E2E2FZZ</td></tr>
<tr><th colspan="3">Snort Logs</th></tr>
<tr><td colspan="3">None available</td></tr>
</table>

Increasing the time further will eventually evade Cisco Secure IDS.

Example 9: 2 Byte splices delayed by 100 seconds

<table>
<tr><th>Request</th><th>Splice Size</th><th>Timing</th></tr>
<tr><td>GET /scripts/..%c0%af../ HTTP/1.0\r\n</td><td>2</td><td>100</td></tr>
<tr><th colspan="3">Cisco IDS Logs</th></tr>
<tr><td colspan="3">None Available</td></tr>
<tr><th colspan="3">Snort Logs</th></tr>
<tr><td colspan="3">None Available</td></tr>
</table>

In basic session splicing attacks where the hosts reside on the same broadcast domain it is possible to evade the IDS by not using the default one byte splice and delaying the attack significantly. The reason different Snort signatures will trigger, which can make an attack seem less threatening is because of Snort's first exit rule. I suspected the problem was in the way rules were passed through the detection engine for comparison. I posted an email to Snort-Devel mail list and received this response from Martin Roesch the founder of Snort.

"If you want the single-byte detects for the real scripts that are being accessed, turn off the rule that's going off, that's Snort's "first exit" engine doing it's job. If you want to extend the tracking time for a session, increase the default timeout value for the stream4 preprocessor:

preprocessor stream4: timeout 3600, detect_scans"

This helps explain why different Snort rules would trigger under different circumstances. I did some basic testing with turning of the session splicing rules and setting the timeout longer. Snort does a better job off detecting the true attack without the evasion signatures. However, the early exit rule still causes a Unicode cmd.exe attack to be seen as scripts access, and other attacks such as .ida to be misdiagnosed depending on how the splices fall and their timing.

Whether these methods are successful when an IDS employs session re-assembly is somewhat host and application dependent. In testing, Apache on RedHat sessions, time out in six minutes while IIS on Windows 2000 doesn't appear to timeout in any reasonable amount of time. This timeout of six minutes by Apache makes time based evasion more difficult when the host is on the same broadcast domain as the victim. To remove the broadcast domain, the web server was given a new IP behind a router one additional hop from the sensors.

Lab Setup 2

Host Type	IP	Version	Function
Snort Sensor	W.X.Y.122	Snort 1.8.6 / RH 7.2	IDS / Apache Web Server
Cisco Secure IDS	W.X.Y.123	3.1 Beta	IDS Only
Windows Host IDS	W.X.Y.108	W2K Server / Entercept 2.01	IIS Web Server HIDS Agent
Cisco 3600 Router	W.X.Y.125	IOS	Router

The victim host and IDS devices are now not on the same broadcast domain. The victim host is also now one extra hop away. To help expedite the test, a simple custom signature was created on both IDS devices to alarm on the URL foo.htm. This was done to avoid triggering any alarms on the Entercept HIDS agent and to avoid any first exit rules.

Test Signature Snort:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS   kevin
foo test access"; flags: A+; uricontent:"foo.htm"; nocase; classtype:web-
application-attack; sid: 1256; rev:2;)
```

Test Signature Cisco Secure IDS # 30001:

```
Engine STATE.HTTP SIGID 30001 AlarmThrottle FireOnce ChokeThreshold
ANY DeObfuscate True Direction ToService MinHits 1 ResetAfterIdle 15
ServicePorts 80,3128,8000,8010,8080,8888,24326 SigName foo ThrottleInterval 15
UriRegex foo.htm
```

To test that re-assembly was working well, and there were no other problems prohibiting proper alarming, a quick test was run sending 6 byte splices quickly.

Both devices alarmed properly without a problem.

Test 1: Session 1 Fast

```
18:36:12.746266 64.194.107.85.33296 > W.X.Y.108.80: S 332613146:332613146(0) win 5840 <mss
1460,sackOK,timestamp 78415572[|tcp]> (DF) (ttl 64, id 49358, len 60)
18:36:12.906266 W.X.Y.108.80 > 64.194.107.85.33296: S 2058514897:2058514897(0) ack 332613147
win 17520 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF) (ttl 116, id 2480, len 64)
18:36:12.906266 64.194.107.85.33296 > W.X.Y.108.80: . [tcp sum ok] ack 1 win 5840
<nop,nop,timestamp 78415588 0> (DF) (ttl 64, id 49359, len 52)
18:36:12.906266 64.194.107.85.33296 > W.X.Y.108.80: P 1:8(7) ack 1 win 5840 <nop,nop,timestamp
78415588 0> (DF) (ttl 64, id 49360, len 59)
18:36:13.146266 W.X.Y.108.80 > 64.194.107.85.33296: . [tcp sum ok] ack 8 win 17513
<nop,nop,timestamp 90556 78415588> (DF) (ttl 116, id 2481, len 52)
18:36:13.906266 64.194.107.85.33296 > W.X.Y.108.80: P 8:14(6) ack 1 win 5840 <nop,nop,timestamp
78415688 90556> (DF) (ttl 64, id 49361, len 58)
18:36:14.156266 W.X.Y.108.80 > 64.194.107.85.33296: . [tcp sum ok] ack 14 win 17507
<nop,nop,timestamp 90566 78415688> (DF) (ttl 116, id 2482, len 52)
18:36:14.906266 64.194.107.85.33296 > W.X.Y.108.80: P 14:20(6) ack 1 win 5840 <nop,nop,timestamp
78415788 90566> (DF) (ttl 64, id 49362, len 58)
18:36:15.156266 W.X.Y.108.80 > 64.194.107.85.33296: . [tcp sum ok] ack 20 win 17501
<nop,nop,timestamp 90576 78415788> (DF) (ttl 116, id 2483, len 52)
18:36:15.906266 64.194.107.85.33296 > W.X.Y.108.80: P 20:71(51) ack 1 win 5840 <nop,nop,timestamp
78415888 90576> (DF) (ttl 64, id 49363, len 103)
18:36:16.156266 W.X.Y.108.80 > 64.194.107.85.33296: . [tcp sum ok] ack 71 win 17450
<nop,nop,timestamp 90586 78415888> (DF) (ttl 116, id 2492, len 52)
18:36:16.446266 W.X.Y.108.80 > 64.194.107.85.33296: . 1:1449(1448) ack 71 win 17450
<nop,nop,timestamp 90587 78415888> (DF) (ttl 116, id 2493, len 1500)
18:36:16.446266 64.194.107.85.33296 > W.X.Y.108.80: . [tcp sum ok] ack 1449 win 8688
<nop,nop,timestamp 78415942 90587> (DF) (ttl 64, id 49364, len 52)
18:36:16.546266 W.X.Y.108.80 > 64.194.107.85.33296: . 1449:2897(1448) ack 71 win 17450
<nop,nop,timestamp 90587 78415888> (DF) (ttl 116, id 2494, len 1500)
18:36:16.546266 W.X.Y.108.80 > 64.194.107.85.33296: . 2897:2921(24) ack 71 win 17450
<nop,nop,timestamp 90588 78415888> (DF) (ttl 116, id 2495, len 76)
18:36:16.546266 64.194.107.85.33296 > W.X.Y.108.80: . [tcp sum ok] ack 2897 win 11584
<nop,nop,timestamp 78415952 90587> (DF) (ttl 64, id 49365, len 52)
18:36:16.546266 64.194.107.85.33296 > W.X.Y.108.80: . [tcp sum ok] ack 2921 win 11584
<nop,nop,timestamp 78415952 90588> (DF) (ttl 64, id 49366, len 52)
18:36:16.576266 W.X.Y.108.80 > 64.194.107.85.33296: FP 2921:3397(476) ack 71 win 17450
<nop,nop,timestamp 90589 78415942> (DF) (ttl 116, id 2496, len 528)
18:36:16.576266 64.194.107.85.33296 > W.X.Y.108.80: F [tcp sum ok] 71:71(0) ack 3398 win 14480
<nop,nop,timestamp 78415955 90589> (DF) (ttl 64, id 49367, len 52)
18:36:16.656266 W.X.Y.108.80 > 64.194.107.85.33296: . [tcp sum ok] ack 72 win 17450
<nop,nop,timestamp 90590 78415955> (DF) (ttl 116, id 2497, len 52)
```

Cisco IDS Logs

```
4,1064599,2002/05/01,00:33:35,2002/04/30,19:33:35,10008,100,101,OUT,IN,1,3000,80,
TCP/IP,64.194.107.85,W.X.Y.108,33296,80,0.0.0.0,332613146
4,1064600,2002/05/01,00:33:36,2002/04/30,19:33:36,10008,100,101,OUT,IN,4,30001,0,
TCP/IP,64.194.107.85,W.X.Y.108,33296,80,0.0.0.0, ,474554202F666F6F2E68746D20ZZ
```

Snort Logs

```
04/30-19:33:46.690000  [**] [1:1256:2] WEB-IIS  kevin foo test access [**] [Classification: Web
Application Attack] [Priority: 1] {TCP} 64.194.107.85:33296 -> W.X.Y.108:80
```

Both the devices alerted on the custom signatures while using basic session splicing techniques. The Cisco device alerted on a signature 3000 which is a TCP connection and signature 30001 which is the custom foo.htm signature. This proves session re-assembly is working correctly on both devices. The following evasion technique takes advantage of inherent network problems as described by Newsham and Ptacek[1] and Vern Paxson and Mark Handley in their paper "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to End Protocol Semantics"[4]. The following test was done.

- Fake resets

To understand the fake reset, the basic principle is to send a reset packet with a low TTL value destined to the host that the IDS will see and understand as a session teardown but will timeout before the host. The next logs are from a fake reset, which Snort is and Cisco are both susceptible to. There are other evasion techniques similar to this one that can be used when a host is more hops away. These techniques for splicing all take advantage of a fake packet timing out before it gets to the host. Using these techniques on can send multiple packets containing the same data, hoping the IDS will use the incorrect data to perform analysis. Only fake reset packets were tested, since all techniques work on the same premise. Note the timing was set a longer timeout to give myself time to craft the reset packet using HPING2. The inserted reset and its associated response (from the router icmp ttl exceeded message) are highlighted in red.

Test 2: Fake Reset Successful

```
18:43:34.316266 64.194.107.85.33298 > W.X.Y.108.80: S 788083420:788083420(0) win 5840 <mss
1460,sackOK,timestamp 78459729[|tcp]> (DF)
18:43:34.386266 W.X.Y.108.80 > 64.194.107.85.33298: S 2169462490:2169462490(0) ack 788083421
win 17520 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
18:43:34.386266 64.194.107.85.33298 > W.X.Y.108.80: . ack 1 win 5840 <nop,nop,timestamp 78459736
0> (DF)
18:43:34.386266 64.194.107.85.33298 > W.X.Y.108.80: P 1:8(7) ack 1 win 5840 <nop,nop,timestamp
78459736 0> (DF)
18:43:34.596266 W.X.Y.108.80 > 64.194.107.85.33298: . ack 8 win 17513 <nop,nop,timestamp 94970
78459736> (DF)
18:43:57.116266 64.194.107.85.33298 > W.X.Y.108.80: R 788083428:788083428(0) win 512
18:43:57.196266 W.X.Y.125 > 64.194.107.85: icmp: time exceeded in-transit [tos 0xc0]
18:44:04.386266 64.194.107.85.33298 > W.X.Y.108.80: P 8:14(6) ack 1 win 5840 <nop,nop,timestamp
78462736 94970> (DF)
18:44:04.636266 W.X.Y.108.80 > 64.194.107.85.33298: . ack 14 win 17507 <nop,nop,timestamp 95271
78462736> (DF)
18:44:34.386266 64.194.107.85.33298 > W.X.Y.108.80: P 14:20(6) ack 1 win 5840 <nop,nop,timestamp
78465736 95271> (DF)
18:44:34.586266 W.X.Y.108.80 > 64.194.107.85.33298: . ack 20 win 17501 <nop,nop,timestamp 95570
78465736> (DF)
18:45:04.386266 64.194.107.85.33298 > W.X.Y.108.80: P 20:71(51) ack 1 win 5840 <nop,nop,timestamp
78468736 95570> (DF)
18:45:04.566266 W.X.Y.108.80 > 64.194.107.85.33298: . 1:1449(1448) ack 71 win 17450
<nop,nop,timestamp 95868 78468736> (DF)
18:45:04.566266 64.194.107.85.33298 > W.X.Y.108.80: . ack 1449 win 8688 <nop,nop,timestamp
```

78468754 95868> (DF) 18:45:04.666266 W.X.Y.108.80 > 64.194.107.85.33298: . 1449:2897(1448) ack 71 win 17450 <nop,nop,timestamp 95868 78468736> (DF) 18:45:04.666266 W.X.Y.108.80 > 64.194.107.85.33298: . 2897:2921(24) ack 71 win 17450 <nop,nop,timestamp 95868 78468736> (DF) 18:45:04.666266 64.194.107.85.33298 > W.X.Y.108.80: . ack 2897 win 11584 <nop,nop,timestamp 78468764 95868> (DF) 18:45:04.666266 64.194.107.85.33298 > W.X.Y.108.80: . ack 2921 win 11584 <nop,nop,timestamp 78468764 95868> (DF) 18:45:04.696266 W.X.Y.108.80 > 64.194.107.85.33298: FP 2921:3397(476) ack 71 win 17450 <nop,nop,timestamp 95870 78468754> (DF) 18:45:04.706266 64.194.107.85.33298 > W.X.Y.108.80: F 71:71(0) ack 3398 win 14480 <nop,nop,timestamp 78468768 95870> (DF) 18:45:04.776266 W.X.Y.108.80 > 64.194.107.85.33298: . ack 72 win 17450 <nop,nop,timestamp 95871 78468768> (DF)
Cisco IDS Logs
4,1064610,2002/05/01,00:40:56,2002/04/30,19:40:56,10008,100,101,OUT,IN,1,3000,80,TCP/IP ,64.194.107.85,W.X.Y.108,33298,80,0.0.0.0,788083420 4,1064611,2002/05/01,00:41:30,2002/04/30,19:41:30,10008,100,101,IN,OUT,1,2005,0,TCP/IP ,W.X.Y.125,64.194.107.85,0,0,0.0.0.0,
Snort Logs
None Available

Both IDS devices were susceptible to this fake reset attack. Snort logged nothing while Cisco Secure IDS only logged a signature 3000 connection request and a 2005 ICMP destination unreachable message. It should be noted that Snort developers are currently working to defeat this style of network evasion through the use of assigning a minimum TTL required for assembly of the session. Much of this has been driven by the recent release of a tool by Dug Song called fragroute[5] which tests many of the fragmentation problems described by Newsham and Ptacek[1].

Conclusion:

Several different session splicing evasion techniques have been demonstrated with varying degrees of success. Most of the techniques that are available with fragmentation can be used to some degree with splicing. The following table summarizes some of the similarities between fragmentation and splicing and denotes whether I had any success in limited testing.

A comparison of fragmentation and session splicing evasion techniques:

Technique	Fragmentation	Session Splicing	Successful in limited testing
Splitting payload	Yes	Yes	Limited
Data overwrite	Yes	Yes	Not tested
Data Insertion	Yes	Yes	Not tested
Fake Reset Teardown	Yes	Yes	Yes
Delayed delivery	Limited	Yes	Yes
Trigger less threatening rules	Yes	Yes	Yes

The reality is that many network level problems still exist and are very difficult for devices to handle without traffic normalization unless the device uses “bifurcating analysis”[4] techniques which means that if an IDS detects traffic which has possible multiple interpretations it will apply all interpretations to the analysis and alarm if any match a signature. Still, the use of session splicing timeouts presents a unique problem for the IDS. If a host operating system will keep the session alive for a very long period of time, than the IDS must do the same. IDS designers recently have made tremendous strides in defeating string matching evasion techniques; now network level techniques are being addressed.

Assignment 2: Network Detects

Detects:

A combination of several networks were used for sourcing detects. A description of the networks is below.

Network #1:
Home network. This is my home network, a DSL connected Red Hat Linux 7.1 host serving as a combination iptables firewall and Snort IDS. Snort is version 1.8.4 using the standard rule set plus a few custom rules. The custom rules will only be referenced if they were an integral part of a detect. The host is running a variety of services including Web, FTP, Telnet, SSH, MySQL, and Portmap. The firewall allows world access to all of these services. Port 81 and 53 TCP are port forwarded to a Windows 2000 server on the internal network. SSH is OpenSSH version 2.5.2p2. SSH has been configured to only allow protocol version 2 and to not use login. Web Ports 80 and 443 are password protected. This host runs a constant version of tcpdump logging everything with the command tcpdump –w /dumpfiles.tcpdump-$date.dmp &.

Network #2:
This network is a Red Hat 7.2 host serving Web, Email and DNS for 81 domains. This host runs Sendmail, Bind, Apache, MySQL, FTP, SSH, IMAP and Portmap. This host has no firewall but uses a combination of Portsentry and tcp-wrappers for protection. This host is running Snort version 1.8.4 using a default rule set with the addition of a few custom rules. The custom rules will only be referenced if they are applicable to a detect. This host runs a constant version of tcpdump logging everything with the command tcpdump –w /dumpfiles.tcpdump-$date.dmp &.

Network #3:
This network is a combination of networks my employer manages. It consists of Cisco Secure IDS sensors, Cisco PIX firewalls, Checkpoint FW1 firewalls, and Entercept HIDS agents. Since these are managed hosts and customer networks IP addresses will be sanitized. Once again the sensors use default configuration with the addition of custom signatures.

Network #4:
This is a lab network consisting of both Snort 1.8.6 and Cisco 3.1IDS sensors. A Windows 2000 web server running IIS with Entercept HIDS, and a Red Hat 6.2 server with Telnet and Apache open. There is not functional firewall in this network. This network also runs a constant version of tcpdump logging everything with the command tcpdump –w /dumpfiles.tcpdump-$date.dmp &.

Detect #1: Noisy port scan followed by session spliced web attacks

Attack overview and analysis:

This detect consisted of 442 Snort alerts. A breakdown of the 442 alerts is as follows. The Source IP is 172.143.143.116.

```
root@server1 snort]# grep 172.143.143.116 alert-3-36  |awk -F] '{print $3}' |cut -c 0-30
|sort |uniq -c |sort -r
```

```
    432  spp_portscan: portscan status
      4  Abnormal Web Request
      2  SCAN Proxy attempt [**
      1  spp_portscan: PORTSCAN DETECT
      1  spp_portscan: End of portscan
      1  Session Splicing attempt [**
      1  INFO - Possible Squid Scan [*
```

Snort alerts without port scan entries

```
03/24-00:53:50.960307  [**] [1:0:0] Abnormal Web Request [**]
[Classification: Potentially Bad Traffic] [Priority: 2] {TCP}
172.143.143.116:2589 -> W.X.Y.16:80
03/24-00:57:53.119547  [**] [1:615:2] SCAN Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
172.143.143.116:3699 -> W.X.Y.16:1080
03/24-01:03:46.370944  [**] [1:618:1] INFO - Possible Squid Scan [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
172.143.143.116:1806 -> W.X.Y.16:3128
03/24-01:17:42.150156  [**] [1:620:1] SCAN Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
172.143.143.116:2873 -> W.X.Y.16:8080
03/24-01:36:50.513678  [**] [1:0:0] Abnormal Web Request [**]
[Classification: Potentially Bad Traffic] [Priority: 2] {TCP}
172.143.143.116:1164 -> W.X.Y.16:80
03/24-01:45:22.697933  [**] [1:0:0] Abnormal Web Request [**]
[Classification: Potentially Bad Traffic] [Priority: 2] {TCP}
172.143.143.116:1182 -> W.X.Y.16:80
03/24-01:45:42.057991  [**] [1:0:0] Abnormal Web Request [**]
[Classification: Potentially Bad Traffic] [Priority: 2] {TCP}
172.143.143.116:1183 -> W.X.Y.16:80
03/24-01:47:18.979763  [**] [1:0:0] Session Splicing attempt [**]
[Classification: Potentially Bad Traffic] [Priority: 2] {TCP}
172.143.143.116:1184 -> W.X.Y.16:80
```

Signatures triggered by this attack.

Abnormal web request

```
local.rules:alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80
(msg:"Abnormal Web Request";
flags:A+;content:"htt";nocase;content:!"accept";nocase;content:!"User-
Agent";nocase; classtype:bad-unknown;)
```

This is a custom rule I wrote. The basic premise behind this rule was to catch web requests not intitiated by a web browser. Most automated web scanning tools do not send requests with the same information as a web browser. This particular rule catches requests that contain htt but do not contain the User-Agent or accept headers. This rule triggers on many of the CodeRed and Concept worm scans.

Scan proxy attempt

```
scan.rules:alert tcp $EXTERNAL_NET any -> $HOME_NET 1080 (msg:"SCAN
Proxy attempt"; flags:S; reference:url,help.undernet.org/proxyscan/;
classtype:attempted-recon; sid:615; rev:2;)
scan.rules:alert tcp $EXTERNAL_NET any -> $HOME_NET 8080 (msg:"SCAN
Proxy attempt";flags:S; classtype:attempted-recon; sid:620; rev:1;)
```

These two rules are from the standard Snort rule set. These rules look for SYN packets to ports 1080 and 8080. These ports are associated with well-known proxies. In this case these rules triggered as part of a larger port scan. The port scan log supports this theory. In this attempt the attacker scanned every port from port 1 to port 9677.

INFO – Possible Squid Scan

```
scan.rules:alert tcp $EXTERNAL_NET any -> $HOME_NET 3128 (msg:"INFO -
Possible Squid Scan"; flags:S; classtype:attempted-recon; sid:618;
rev:1;)
```

This rule is from the standard Snort rule set. This triggers on SYN requests to port 3128. This is part of the larger port scan.

Session Splicing attempt

```
local.rules:alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80
(msg:"Session Splicing attempt"; flags:A+;content:"g";nocase; dsize:
<12; classtype:bad-unknown;)
```

This is a custom rule I wrote as well. Snort does have two rules to catch Whisker session splicing attempts. I feel that these rules are a little to specific to Whisker and miss other tools. This rule looks for a "g" in a web request and a total content length of less than 13 bytes. Normal web request are formatted as "GET HTTP/1.0\r\n\r\n" require 16 bytes in the get request. This rule benignly triggers often, and has its own deficiencies

Here is the output from the Snort logs. This gives us a better view into the packets that triggered the specific alerts.

Snort Logs:

```
-*> Snort! <*-
Version 1.8.4 (Build 99)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
03/24-00:53:50.960307 172.143.143.116:2589 -> W.X.Y.16:80
TCP TTL:114 TOS:0x0 ID:33060 IpLen:20 DgmLen:120 DF
***AP*** Seq: 0xA52A7565  Ack: 0x5A6E561B  Win: 0x4322  TcpLen: 20
```

```
47 45 54 20 2F 20 48 54 54 50 2F 31 2E 30 0D 0A  GET / HTTP/1.0..
43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 43 6C 6F 73  Connection: Clos
65 0D 0A 50 72 61 67 6D 61 3A 20 6E 6F 2D 63 61  e..Pragma: no-ca
63 68 65 0D 0A 43 6F 6E 74 65 6E 74 2D 54 79 70  che..Content-Typ
65 3A 20 74 65 78 74 2F 68 74 6D 6C 0D 0A 0D 0A  e: text/html....

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

03/24-00:57:53.119547 172.143.143.116:3699 -> W.X.Y.16:1080
TCP TTL:114 TOS:0x0 ID:36997 IpLen:20 DgmLen:48 DF
******S* Seq: 0xAC00C9DE  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1322 NOP NOP SackOK

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

03/24-01:03:46.370944 172.143.143.116:1806 -> W.X.Y.16:3128
TCP TTL:114 TOS:0x0 ID:43598 IpLen:20 DgmLen:48 DF
******S* Seq: 0xB75D679F  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1322 NOP NOP SackOK

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

03/24-01:17:42.150156 172.143.143.116:2873 -> W.X.Y.16:8080
TCP TTL:114 TOS:0x0 ID:59633 IpLen:20 DgmLen:48 DF
******S* Seq: 0xD29213CD  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1322 NOP NOP SackOK

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

03/24-01:36:50.513678 172.143.143.116:1164 -> W.X.Y.16:80
TCP TTL:114 TOS:0x0 ID:2782 IpLen:20 DgmLen:51 DF
***AP*** Seq: 0xEA4473BF  Ack: 0xFD3820BE  Win: 0x4322  TcpLen: 20
68 74 74 70 64 2E 63 6F 6E 66 20                 httpd.conf

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

03/24-01:45:22.697933 172.143.143.116:1182 -> W.X.Y.16:80
TCP TTL:114 TOS:0x0 ID:3119 IpLen:20 DgmLen:55 DF
***AP*** Seq: 0xF1F152B0  Ack: 0x1CD6D5DC  Win: 0x4322  TcpLen: 20
69 6E 64 65 78 2E 68 74 6D 6C 20 48 54 54 50     index.html HTTP

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

03/24-01:45:42.057991 172.143.143.116:1183 -> W.X.Y.16:80
TCP TTL:114 TOS:0x0 ID:3161 IpLen:20 DgmLen:64 DF
***AP*** Seq: 0xF23D144B  Ack: 0x1DF5A5E1  Win: 0x4322  TcpLen: 20
47 45 54 20 2F 69 6E 64 65 78 2E 68 74 6D 6C 20  GET /index.html
48 54 54 50 2F 31 2E 30                          HTTP/1.0

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

03/24-01:47:18.979763 172.143.143.116:1184 -> W.X.Y.16:80
TCP TTL:114 TOS:0x0 ID:3196 IpLen:20 DgmLen:51 DF
***AP*** Seq: 0xF3B0519C  Ack: 0x251BD1A1  Win: 0x4322  TcpLen: 20
63 6F 6E 66 69 67 5F 6C 6F 67 5F                 config_log_
```

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

The session splicing log is the last packet. We will use tcpdump http://tcpdump.org logs to view these and logs and determine the attackers motives.

Tcpdump logs of Snort alert

```
01:47:17.179764 172.143.143.116.1184 > W.X.Y.16.http: S 4088418715:4088418715(0) win 16384 <mss
1322,nop,nop,sackOK> (DF)
01:47:17.180785 W.X.Y.16.http > 172.143.143.116.1184: S 622580128:622580128(0) ack 4088418716
win 5840 <mss 1460,nop,nop,sackOK> (DF)
01:47:17.384921 172.143.143.116.1184 > W.X.Y.16.http: . ack 1 win 17186 (DF)
01:47:18.979763 172.143.143.116.1184 > W.X.Y.16.http: P 1:12(11) ack 1 win 17186 (DF)
01:47:18.979857 W.X.Y.16.http > 172.143.143.116.1184: . ack 12 win 5840 (DF)
01:47:19.729700 172.143.143.116.1184 > W.X.Y.16.http: P 12:14(2) ack 1 win 17186 (DF)
01:47:19.729788 W.X.Y.16.http > 172.143.143.116.1184: . ack 14 win 5840 (DF)
01:47:19.731023 W.X.Y.16.http > 172.143.143.116.1184: P 1:333(332) ack 14 win 5840 (DF)
01:47:19.731214 W.X.Y.16.http > 172.143.143.116.1184: F 333:333(0) ack 14 win 5840 (DF)
01:47:19.982073 172.143.143.116.1184 > W.X.Y.16.http: . ack 334 win 16854 (DF)
01:47:21.522640 172.143.143.116.1184 > W.X.Y.16.http: F 14:14(0) ack 334 win 16854 (DF)
01:47:21.522683 W.X.Y.16.http > 172.143.143.116.1184: . ack 15 win 5840 (DF)
```

The target server responded to this attempt with an HTTP 501 Method not Implemented response.

```
01:47:19.731023 W.X.Y.16.http > 172.143.143.116.1184: P [tcp sum ok] 1:333(332) ack 14 win 5840
(DF) (ttl 64, id 22265, len 372)
0x0000   4500 0174 56f9 4000 4006 5550 4027 1110       E..tV.@.@.UP@'..
0x0010   ac8f 8f74 0050 04a0 251b d1a1 f3b0 51a9       ...t.P..%.....Q.
0x0020   5018 16d0 f643 0000 3c21 444f 4354 5950       P....C..<!DOCTYP
0x0030   4520 4854 4d4c 2050 5542 4c49 4320 222d       E.HTML.PUBLIC."-
0x0040   2f2f 4945 5446 2f2f 4454 4420 4854 4d4c       //IETF//DTD.HTML
0x0050   2032 2e30 2f2f 454e 223e 0a3c 4854 4d4c       .2.0//EN">.<HTML
0x0060   3e3c 4845 4144 3e0a 3c54 4954 4c45 3e35       ><HEAD>.<TITLE>5
0x0070   3031 204d 6574 686f 6420 4e6f 7420 496d       01.Method.Not.Im
0x0080   706c 656d 656e 7465 643c 2f54 4954 4c45       plemented</TITLE
0x0090   3e0a 3c2f 4845 4144 3e3c 424f 4459 3e0a       >.</HEAD><BODY>.
0x00a0   3c48 313e 4d65 7468 6f64 204e 6f74 2049       <H1>Method.Not.I
0x00b0   6d70 6c65 6d65 6e74 6564 3c2f 4831 3e0a       mplemented</H1>.
0x00c0   636f 6e66 6967 5f6c 6f67 5f20 746f 202f       config_log_.to./
0x00d0   696e 6465 782e 6874 6d20 6e6f 7420 7375       index.htm.not.su
0x00e0   7070 6f72 7465 642e 3c50 3e0a 496e 7661       pported.<P>.Inva
0x00f0   6c69 6420 6d65 7468 6f64 2069 6e20 7265       lid.method.in.re
0x0100   7175 6573 7420 636f 6e66 6967 5f6c 6f67       quest.config_log
0x0110   5f3c 503e 0a3c 4852 3e0a 3c41 4444 5245       _<P>.<HR>.<ADDRE
0x0120   5353 3e41 7061 6368 652f 312e 332e 3232       SS>Apache/1.3.22
```

The tcpdump logs show evidence of a true session splicing attempt that was not detected by Snort. This attempt did not elicit a response from the server.

```
tcpdump -r 172.143.143.116.dmp -Xvn "(src port 80 and dst port 3521) or (dst port 80
and src port 3521)"
```

```
01:08:27.926211 172.143.143.116.3521 > W.X.Y.16.http: S [tcp sum ok] 3231429409:3231429409(0) win
16384 <mss 1322,nop,nop,sackOK> (DF) (ttl 114, id 48999, len 48)
0x0000   4500 0030 bf67 4000 7206 bc25 ac8f 8f74        E..0.g@.r..%...t
0x0010   4027 1110 0dc1 0050 c09b b321 0000 0000        @'.....P...!....
0x0020   7002 4000 34a0 0000 0204 052a 0101 0402        p.@.4......*....
01:08:27.926266 W.X.Y.16.http > 172.143.143.116.3521: S [tcp sum ok] 2426323710:2426323710(0) ack
3231429410 win 5840 <mss 1460,nop,nop,sackOK> (DF) (ttl 64, id 0, len 48)
0x0000   4500 0030 0000 4000 4006 ad8d 4027 1110        E..0..@.@...@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2fe c09b b322        ...t.P........."
0x0020   7012 16d0 0998 0000 0204 05b4 0101 0402        p...............
01:08:28.135382 172.143.143.116.3521 > W.X.Y.16.http: . [tcp sum ok] ack 1 win 17186 (DF) (ttl 114, id
49004, len 40)
0x0000   4500 0028 bf6c 4000 7206 bc28 ac8f 8f74        E..(.l@.r..(...t
0x0010   4027 1110 0dc1 0050 c09b b322 909e c2ff        @'.....P..."....
0x0020   5010 4322 0a0a 0000 0000 0000 0000             P.C"..........
01:08:29.092131 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 1:2(1) ack 1 win 17186 (DF) (ttl
114, id 49022, len 41)
0x0000   4500 0029 bf7e 4000 7206 bc15 ac8f 8f74        E..).~@.r......t
0x0010   4027 1110 0dc1 0050 c09b b322 909e c2ff        @'.....P..."....
0x0020   5018 4322 a600 0000 6400 0000 0000             P.C"....d.....
01:08:29.092212 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 2 win 5840 (DF) (ttl 64, id
15196, len 40)
0x0000   4500 0028 3b5c 4000 4006 7239 4027 1110        E..(;\@.@.r9@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b323        ...t.P.........#
0x0020   5010 16d0 365b 0000                            P...6[..
01:08:29.345298 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 2:3(1) ack 1 win 17186 (DF) (ttl
114, id 49027, len 41)
0x0000   4500 0029 bf83 4000 7206 bc10 ac8f 8f74        E..)..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b323 909e c2ff        @'.....P...#....
0x0020   5018 4322 97ff 0000 7200 0000 0000             P.C"....r.....
01:08:29.345379 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 3 win 5840 (DF) (ttl 64, id
15197, len 40)
0x0000   4500 0028 3b5d 4000 4006 7238 4027 1110        E..(;]@.@.r8@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b324        ...t.P.........$
0x0020   5010 16d0 365a 0000                            P...6Z..
01:08:29.562648 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 3:4(1) ack 1 win 17186 (DF) (ttl
114, id 49030, len 41)
0x0000   4500 0029 bf86 4000 7206 bc0d ac8f 8f74        E..)..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b324 909e c2ff        @'.....P...$....
0x0020   5018 4322 94fe 0000 7500 0000 0000             P.C"....u.....
01:08:29.562717 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 4 win 5840 (DF) (ttl 64, id
15198, len 40)
0x0000   4500 0028 3b5e 4000 4006 7237 4027 1110        E..(;^@.@.r7@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b325        ...t.P.........%
0x0020   5010 16d0 3659 0000                            P...6Y..
01:08:30.040547 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 4:5(1) ack 1 win 17186 (DF) (ttl
114, id 49042, len 41)
0x0000   4500 0029 bf92 4000 7206 bc01 ac8f 8f74        E..)..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b325 909e c2ff        @'.....P...%....
0x0020   5018 4322 a2fd 0000 6700 0000 0000             P.C"....g.....
01:08:30.040622 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 5 win 5840 (DF) (ttl 64, id
15199, len 40)
0x0000   4500 0028 3b5f 4000 4006 7236 4027 1110        E..(;_@.@.r6@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b326        ...t.P.........&
0x0020   5010 16d0 3658 0000                            P...6X..
01:08:30.739975 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 5:6(1) ack 1 win 17186 (DF) (ttl
114, id 49055, len 41)
0x0000   4500 0029 bf9f 4000 7206 bbf4 ac8f 8f74        E..)..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b326 909e c2ff        @'.....P...&....
0x0020   5018 4322 a1fc 0000 6800 0000 0000             P.C"....h.....
01:08:30.740060 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 6 win 5840 (DF) (ttl 64, id
15200, len 40)
0x0000   4500 0028 3b60 4000 4006 7235 4027 1110        E..(;`@.@.r5@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b327        ...t.P.........'
0x0020   5010 16d0 3657 0000                            P...6W..
01:08:31.302186 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 6:7(1) ack 1 win 17186 (DF) (ttl
114, id 49066, len 41)
0x0000   4500 0029 bfaa 4000 7206 bbe9 ac8f 8f74        E..)..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b327 909e c2ff        @'.....P...'....
0x0020   5018 4322 9afb 0000 6f00 0000 0000             P.C"....o.....
01:08:31.302271 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 7 win 5840 (DF) (ttl 64, id
15201, len 40)
0x0000   4500 0028 3b61 4000 4006 7234 4027 1110        E..(;a@.@.r4@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b328        ...t.P.........(
0x0020   5010 16d0 3656 0000                            P...6V..
01:08:31.637278 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 7:8(1) ack 1 win 17186 (DF) (ttl
114, id 49073, len 41)
0x0000   4500 0029 bfb1 4000 7206 bbe2 ac8f 8f74        E..)..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b328 909e c2ff        @'.....P...(....
0x0020   5018 4322 a4fa 0000 6500 0000 0000             P.C"....e.....
01:08:31.637370 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 8 win 5840 (DF) (ttl 64, id
15202, len 40)
0x0000   4500 0028 3b62 4000 4006 7233 4027 1110        E..(;b@.@.r3@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b329        ...t.P.........)
0x0020   5010 16d0 3655 0000                            P...6U..
01:08:32.052579 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 8:9(1) ack 1 win 17186 (DF) (ttl
114, id 49085, len 41)
0x0000   4500 0029 bfbd 4000 7206 bbd6 ac8f 8f74        E..)..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b329 909e c2ff        @'.....P...)....
0x0020   5018 4322 01fa 0000 0800 0000 0000             P.C"..........
01:08:32.052662 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 9 win 5840 (DF) (ttl 64, id
15203, len 40)
0x0000   4500 0028 3b63 4000 4006 7232 4027 1110        E..(;c@.@.r2@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b32a        ...t.P.........*
0x0020   5010 16d0 3654 0000                            P...6T..
01:08:32.325991 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 9:10(1) ack 1 win 17186 (DF) (ttl
114, id 49092, len 41)
0x0000   4500 0029 bfc4 4000 7206 bbcf ac8f 8f74        E..)..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b32a 909e c2ff        @'.....P...*....
```

```
0x0020   5018 4322 9bf8 0000 6e00 0000 0000           P.C"....n.....
01:08:32.326076 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 10 win 5840 (DF) (ttl 64, id
15204, len 40)
0x0000   4500 0028 3b64 4000 4006 7231 4027 1110       E..(;d@.@.r1@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b32b       ...t.P.........+
0x0020   5010 16d0 3653 0000                           P...6S..
01:08:32.517543 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 10:11(1) ack 1 win 17186 (DF)
(ttl 114, id 49098, len 41)
0x0000   4500 0029 bfca 4000 7206 bbc9 ac8f 8f74       E..)..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b32b 909e c2ff       @'.....P...+....
0x0020   5018 4322 a4f7 0000 6500 0000 0000           P.C"....e.....
01:08:32.517616 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 11 win 5840 (DF) (ttl 64, id
15205, len 40)
0x0000   4500 0028 3b65 4000 4006 7230 4027 1110       E..(;e@.@.r0@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b32c       ...t.P.........,
0x0020   5010 16d0 3652 0000                           P...6R..
01:08:32.800280 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 11:12(1) ack 1 win 17186 (DF)
(ttl 114, id 49108, len 41)
0x0000   4500 0029 bfd4 4000 7206 bbbf ac8f 8f74       E..)..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b32c 909e c2ff       @'.....P...,....
0x0020   5018 4322 90f6 0000 7900 0000 0000           P.C"....y.....
01:08:32.800351 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 12 win 5840 (DF) (ttl 64, id
15206, len 40)
0x0000   4500 0028 3b66 4000 4006 722f 4027 1110       E..(;f@.@.r/@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b32d       ...t.P.........-
0x0020   5010 16d0 3651 0000                           P...6Q..
01:08:33.057797 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 12:13(1) ack 1 win 17186 (DF)
(ttl 114, id 49116, len 41)
0x0000   4500 0029 bfdc 4000 7206 bbb7 ac8f 8f74       E..)..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b32d 909e c2ff       @'.....P...-....
0x0020   5018 4322 dbf5 0000 2e00 0000 0000           P.C"..........
01:08:33.057862 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 13 win 5840 (DF) (ttl 64, id
15207, len 40)
0x0000   4500 0028 3b67 4000 4006 722e 4027 1110       E..(;g@.@.r.@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b32e       ...t.P..........
0x0020   5010 16d0 3650 0000                           P...6P..
01:08:33.375588 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 13:14(1) ack 1 win 17186 (DF)
(ttl 114, id 49127, len 41)
0x0000   4500 0029 bfe7 4000 7206 bbac ac8f 8f74       E..)..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b32e 909e c2ff       @'.....P........
0x0020   5018 4322 a6f4 0000 6300 0000 0000           P.C"....c.....
01:08:33.375657 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 14 win 5840 (DF) (ttl 64, id
15208, len 40)
0x0000   4500 0028 3b68 4000 4006 722d 4027 1110       E..(;h@.@.r-@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b32f       ...t.P........./
0x0020   5010 16d0 364f 0000                           P...6O..
01:08:33.611923 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 14:15(1) ack 1 win 17186 (DF)
(ttl 114, id 49138, len 41)
0x0000   4500 0029 bff2 4000 7206 bba1 ac8f 8f74       E..)..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b32f 909e c2ff       @'.....P.../....
0x0020   5018 4322 9af3 0000 6f00 0000 0000           P.C"....o.....
01:08:33.612008 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 15 win 5840 (DF) (ttl 64, id
15209, len 40)
0x0000   4500 0028 3b69 4000 4006 722c 4027 1110       E..(;i@.@.r,@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b330       ...t.P.........0
0x0020   5010 16d0 364e 0000                           P...6N..
01:08:33.819904 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 15:17(2) ack 1 win 17186 (DF)
(ttl 114, id 49142, len 42)
0x0000   4500 002a bff6 4000 7206 bb9c ac8f 8f74       E..*..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b330 909e c2ff       @'.....P...0....
0x0020   5018 4322 9c83 0000 6d6e 0000 0000           P.C"....mn....
01:08:33.819970 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 17 win 5840 (DF) (ttl 64, id
15210, len 40)
0x0000   4500 0028 3b6a 4000 4006 722b 4027 1110       E..(;j@.@.r+@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b332       ...t.P.........2
0x0020   5010 16d0 364c 0000                           P...6L..
01:08:34.277646 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 17:18(1) ack 1 win 17186 (DF)
(ttl 114, id 49151, len 41)
0x0000   4500 0029 bfff 4000 7206 bb94 ac8f 8f74       E..)..@.r......t
0x0010   4027 1110 0dc1 0050 c09b b332 909e c2ff       @'.....P...2....
0x0020   5018 4322 01f1 0000 0800 0000 0000           P.C"..........
01:08:34.277733 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 18 win 5840 (DF) (ttl 64, id
15211, len 40)
0x0000   4500 0028 3b6b 4000 4006 722a 4027 1110       E..(;k@.@.r*@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b333       ...t.P.........3
0x0020   5010 16d0 364b 0000                           P...6K..
01:08:35.452825 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 18:20(2) ack 1 win 17186 (DF)
(ttl 114, id 49174, len 42)
0x0000   4500 002a c016 4000 7206 bb7c ac8f 8f74       E..*..@.r..|...t
0x0010   4027 1110 0dc1 0050 c09b b333 909e c2ff       @'.....P...3....
0x0020   5018 4322 fce4 0000 0d0a 0000 0000           P.C"..........
01:08:35.452910 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 20 win 5840 (DF) (ttl 64, id
15212, len 40)
0x0000   4500 0028 3b6c 4000 4006 7229 4027 1110       E..(;l@.@.r)@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b335       ...t.P.........5
0x0020   5010 16d0 3649 0000                           P...6I..
01:08:35.454122 W.X.Y.16.http > 172.143.143.116.3521: P [tcp sum ok] 1:345(344) ack 20 win 5840
(DF) (ttl 64, id 15213, len 384)
0x0000   4500 0180 3b6d 4000 4006 70d0 4027 1110       E...;m@.@.p.@'..
0x0010   ac8f 8f74 0050 0dc1 909e c2ff c09b b335       ...t.P.........5
0x0020   5018 16d0 645a 0000 3c21 444f 4354 5950       P...dZ..<!DOCTYP
0x0030   4520 4854 4d4c 2050 5542 4c49 4320 222d       E.HTML.PUBLIC."-
0x0040   2f2f 4945 5446 2f2f 4454 4420 4854 4d4c       //IETF//DTD.HTML
0x0050   2032 2e30 2f2f 454e 223e 0a3c 4854 4d4c       .2.0//EN">.<HTML
0x0060   3e3c 4845 4144 3e0a 3c54 4954 4c45 3e35       ><HEAD>.<TITLE>5
0x0070   3031 204d 6574 686f 6420 4e6f 7420 496d       01.Method.Not.Im
0x0080   706c 656d 656e 7465 643c 2f54 4954 4c45       plemented</TITLE
0x0090   3e0a 3c2f 4845 4144 3e3c 424f 4459 3e0a       >.</HEAD><BODY>.
0x00a0   3c48 313e 4d65 7468 6f64 204e 6f74 2049       <H1>Method.Not.I
0x00b0   6d70 6c65 6d65 6e74 6564 3c2f 4831 3e0a       mplemented</H1>.
0x00c0   6472 7567 686f 6508 6e65 792e 636f 6d6e       drughoe.ney.comn
0x00d0   0820 746f 202f 696e 6465 782e 6874 6d20       ..to./index.htm.
0x00e0   6e6f 7420 7375 7070 6f72 7465 642e 3c50       not.supported.<P
0x00f0   3e0a 496e 7661 6c69 6420 6d65 7468 6f64       >.Invalid.method
0x0100   2069 6e20 7265 7175 6573 7420 6472 7567       .in.request.drug
0x0110   686f 6508 6e65 792e 636f 6d6e 083c 503e       hoe.ney.comn.<P>
0x0120   0a3c 4852 3e0a 3c41 4444 5245 5353 3e41       .<HR>.<ADDRESS>A
0x0130   7061 6368 652f 312e 332e 3232 2053 6572       pache/1.3.22.Ser
```

```
0x0140   7665 7220 6174 2073 6f75 7468 7465 7861        ver.at.southtexa
0x0150   7373 6f6c 7574 696f 6e73 2e63 6f6d 2050       ssolutions.com.P
```

Decoding the actual request reveals the following:

Source IP 172.143.143.116
Source Port 3521
Destination IP W.X.Y.16
Destination Port 80
Request drughoe.ney.comn
Response 501 Method Not Implemented

The same attempt again was made again from several different source ports to port 80. The command used to find these requests is

```
[root@server1 dumpfiles]# tcpdump -r 172.143.143.116.dmp -n '(dst port 80 and tcp[13]
& 0x03 =0) and (len <=61)' |wc -l
   286
```

Decoding each request will reveal many poorly formed Unix style commands. These were all from spliced sessions.

Request and Port Combinations

Port	Request
1161	Log
1162	Files
1163	Rename
1164	Httpd.conf
1165	Srm.xc..conf
1166	Httpd.conf
1167	Access.conf
1168	.htaccess
1169	Httpd
1170	Cog.nf
1171	/etc/inte..etd.conf
1179	Help
1492	&lt:head:&gt
1357	443
3661	W.X.Y.16

Detect #1 Answer section

Source of the attack: Network #2

Detect was generated by:
Snort IDS. With the default Snort rule set this would have looked like a noisy port scan. With the custom rules I created it merited more inspection. Additional logs were provided by tcpdump running in the background logging everything.

Probability the source address was spoofed:
Unlikely, the attack used TCP which requires a three way handshake. The web requests require a completed three way handshake. A lookup of the address shows this as an AOL address

Name Resolution: AC8F8F74.ipt.aol.com.
Trace route distance: 16 hops. The TTL of 114 would be close to a correct windows TTL of 128.
By doing passive OS fingerprinting we realize this is a Windows 2000 host. This finding is based on the facts.

- The base TTL is 128
- The initial window size is 16284
- The IP length of the SYN packet is 48
- TCP Options are MSS, Sack OK and 2 NO-OPS.

This matches the findings of Toby Miller[6] in his paper "Passive OS Fingerprinting: Details and Techniques" http://www.incidents.org/papers/OSfingerprinting.php.

Description of the attack:
This was a strange attack. First the attacker did a very noisy sequential port scan of ports ranging from port 1 to port 9677. Every IDS will alert on this. The attacker then tried to sneak under the radar of an IDS by using session splicing techniques in web requests. The web requests were mostly malformed, and some were misspelled. It makes me believe the attacker was very amateurish but using a possibly advanced tool for the web requests. Of particular interest is the fact that the attack appears to have been run from a Windows host. Most of the tools that perform network evasion tricks are usually Unix based. Whisker which is available at http://www.wiretrip.net/rfp/p/doc.asp/i6/d21.htm has similar abilities and can run on Windows. However, the request are poorly formed and not consistent with default Whisker requests. Whisker behavior can be changed, but if the attacker was skilled enough to modify Whisker I would think they would generate more legitimate and useful requests.

Attack Mechanism:
This reconnaissance portion of this attack was successful. Some of the malformed web requests suggest that the attacker was targeting a Unix OS. One of the session spliced web requests is for drughoe.ney.comn. At the targeted W.X.Y.16 address there is a web sight called drughoney.com. Some of the other web requests seem to target typical Linux files such as /etc/inetd.conf and several well known Apache configuration files. The

attack was unsuccessful as the web server consistently returned "501 Method Not Implemented" error pages. I feel the attack was probably a good tool in novice hands.

Correlations:
I could not find any information on similar attacks.

Evidence of Active Targeting:
There is evidence of active targeting due to the web requests bearing a resemblance to a sight that resides at that IP.

Severity: 1

Criticality - 5 This server is a DNS server, email server and Web server for 81 sites.

Lethality – 1 This attack was very unlikely to succeed against this system.

System Countermeasures – 5 This system is modern system running current software, tcp wrappers and ssh.

Network Countermeasures – 0 There are no network countermeasures. This server resides in a data center with direct connection to the internet.

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

: (5 + 1) – (5 + 0) = 1

Defensive Recommendations: No defensive recommendations are needed for this specific attack. I do feel however the methods used in this attack do present a significant threat, especially if advanced tools are in novice hands on Windows based platforms.

Multiple Choice Test Question:
The following trace could possibly be an example of ?

```
01:08:29.345379 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 3 win 5840 (DF) (ttl 64, id
15197, len 40)
01:08:29.562648 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 3:4(1) ack 1 win 17186 (DF) (ttl
114, id 49030, len 41)
01:08:29.562717 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 4 win 5840 (DF) (ttl 64, id
15198, len 40)
01:08:30.040547 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 4:5(1) ack 1 win 17186 (DF) (ttl
114, id 49042, len 41)
01:08:30.040622 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 5 win 5840 (DF) (ttl 64, id
15199, len 40)
```

```
01:08:30.739975 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 5:6(1) ack 1 win 17186 (DF) (ttl
114, id 49055, len 41)
01:08:30.740060 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 6 win 5840 (DF) (ttl 64, id
15200, len 40)
01:08:31.302186 172.143.143.116.3521 > W.X.Y.16.http: P [tcp sum ok] 6:7(1) ack 1 win 17186 (DF) (ttl
114, id 49066, len 41)
01:08:31.302271 W.X.Y.16.http > 172.143.143.116.3521: . [tcp sum ok] ack 7 win 5840 (DF) (ttl 64, id
15201, len 40)
```

A., A normal web conversation between two Unix hosts.
B., An IIS backdoor scan attempt
C., An attempt at IDS evasion
D., Malformed fragmented packets

Answer – C is the correct answer. These small request passing a byte at a time are an attempt to take advantage of session reconstruction weaknesses of several ID systems.

Detect # 2: SQL Insertion attack on an IIS web server

This detect is a SQL insertion attack. This detect was detected by a Cisco Secure IDS sensor software version 2.2.18. Since this attack was sourced from a corporate network the destination of this attack will be scrubbed from the logs. The signature used to detect this was a custom written signature.

Signature SQL insertion

```
RecordOfStringName  8010   80    1     1
".*=.*[Ss][Ee][Ll][Ee][Cc][Tt].*[Ff][Rr][Oo][Mm]"
```

This signature is a Regex enabled signature designed to look for any HTTP traffic to port 80 with content that includes select and from statements. Select and from statements are normally associated with SQL queries, but they also occur in normal traffic, especially in database driven web sites. This signature admittedly is not perfect which will be discussed later. The following logs are associated with this signature.

Cisco IDS logs

```
4,1403640,2002/03/08,17:51:01,2002/03/08,09:51:01,10008,X,X,OUT,IN,4,8000,8010,
TCP/IP,198.83.130.39,X.X.X.X,2223,80,0.0.0.0,.*=.*[Ss][Ee][Ll][Ee][Cc][Tt].*[Ff][Rr]
[Oo][Mm],67652F6769662C20696D6167652F782D786269746D61702C20696D616765
2F6A7065672C20696D6167652F706A7065672C202A2F2A0D0A4163636570742D4C
616E67756167653A20656E2D75730D0A526566657265723A20687474703A2F2F7777
772E67756573732E636F6D2F7369676E696E2F64656661756C742E6173700D0A5072
61676D613A206E6F2D63616368650D0A436F6F6B69653A2041535053455353494F4E
49445147475147 4F53513D4B47464541424F424E4B4A4347414A44434E424D4D4B4
D460D0A0D0A737465703D7369676E496E26757365726E616D653D2532372B756E69
6F6E2B616C6C2B73656C6563742B6F746865722B66726F6DZZ
4,1403641,2002/03/08,17:51:01,2002/03/08,09:51:01,10008,X,X,OUT,IN,4,8000,8010,
TCP/IP,198.83.130.39,X.X.X.X,2223,80,0.0.0.0,.*=.*[Ss][Ee][Ll][Ee][Cc][Tt].*[Ff][Rr]
[Oo][Mm],20656E2D75730D0A526566657265723A20687474703A2F2F7777772E677
56573732E636F6D2F7369676E696E2F64656661756C742E6173700D0A507261676D
613A206E6F2D63616368650D0A436F6F6B69653A2041535053455353494F4E4944
```

```
51474751474F53513D4B47464541424F424E4B4A4347414A44434E424D4D4B4D46
0D0A0D0A737465703D7369676E496E26757365726E616D653D2532372B756E696F
6E2B616C6C2B73656C6563742B6F746865722B66726F6D2B6F746865727461626C
652B77686572652B253237253237253344253237267061737376F72643D2532372B7
56E696F6E2B616C6C2B73656C6563742B6F746865722B66726F6DZZ
3,1403642,2002/03/08,17:51:05,2002/03/08,09:51:05,10003,X,X,10008,1,2003,EXEC ShunHost
198.83.130.39 1440
3,1403643,2002/03/08,17:51:05,2002/03/08,09:51:05,10003,X,X,10008,1,2003,EXEC ShunHost
198.83.130.39 1440
```

The Cisco IDS event logs are in a comma format of twenty one fields. The following is a description of the fields with the associated value from the first log.

Cisco Log Description

Field	Description	Value
1	Record Type	4
2	Record ID	1403640
3	GMT Datestamp	2002/03/08
4	GMT Timestamp	17:51:01
5	Local Datestamp	2002/03/08
6	Local Timestamp	9:51:01
7	Application ID	10008
8	Host ID	X
9	Organization ID	X
10	Source Direction	OUT
11	Destination Direction	IN
12	Alarm Level	4
13	Sig ID	8000
14	SubSig ID	8010
15	Protocol	TCP/IP
16	Source IP	198.83.130.39
17	Destination IP	X.X.X.X
18	Source Port	2223
19	Destination Port	80
20	Router IP	0.0.0.0
21	Data	

To decode the hex payload we run a perl script to convert the hex to ascii.

Hexdecode Output

```
4,1403640,2002/03/08,17:51:01,2002/03/08,09:51:01,10008,X,X,OUT,IN,4,8000,8010,
TCP/IP,198.83.130.39,X.X.X.X,2223,80,0.0.0.0,.*=.*[Ss][Ee][Ll][Ee][Cc][Tt].*[Ff]
[Rr][Oo][Mm],
ge/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Language: en-us
```

```
Referer: http://www.X.com/signin/X.asp
Pragma: no-cache
Cookie:
step=signIn&username=%27+union+all+select+other+from3


4,1403641,2002/03/08,17:51:01,2002/03/08,09:51:01,10008,X,X,OUT,IN,4,8000,8010,
TCP/IP,198.83.130.39,X.X.X.X,2223,80,0.0.0.0,.*=.*[Ss][Ee][Ll][Ee][Cc][Tt].*[Ff][Rr]
[Oo][Mm],
 en-us
Referer: http://www.X.com/signin/X.asp
Pragma: no-cache
Cookie:
step=signIn&username=%27+union+all+select+other+from+othertable+where+%27
%27%3D%27&password=%27+union+all+select+other+from3
ge/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Language: en-us
Referer: http://www.X.com/signin/X.asp
Pragma: no-cache
Cookie:
```

The attacker formatted two requests with the intent to use SQL injection techniques with malicious intent.

Source of the attack: Network #3

Attack was generated by:
Cisco Secure IDS running version 2.2.18 on a corporate network. This detect was captured through the use of a custom string match signature.

Probability the source address was spoofed:
It is very unlikely that this attack was spoofed. The attack was using an established TCP session that requires a three way handshake. Running nslookup on the IP returns no information, but a whois query reveals that the owner of the address space is New York based ANS communications.

```
[geektools.com]
Query:     198.83.130.39
Registry:  whois.arin.net
Results:
ANS Communications, Inc (NETBLK-BLK198-16-ANS)
   100 Clearbrook Road
   Elmsford, NY 10523
   US

   Netname: BLK198-16-ANS
   Netblock: 198.83.0.0 - 198.83.255.255
   Maintainer: ANS
```

```
Coordinator:
   ANS CO+RE Systems, Inc.  (ANS-NOC-ARIN)  noc@ANS.NET
   1-800-456-6300

Domain System inverse mapping provided by:

NS.ANS.NET                199.171.54.35
NIS.ANS.NET               147.225.1.2

Record last updated on 25-Sep-2001.
Database last updated on  4-Apr-2002 19:59:13 EDT.
```

Description of the attack:
This attack is becoming more commonplace and presents a serious threat to corporate database driven web sites. These attacks require an interactive session by the attacker so there is little chance of this attack being used as a worm. The problems this attack presents are many. The potential for monetary loss or the disclosure of sensitive information is great. These attacks can also be very difficult to defend against because they are not as much an application problem as they are a result of poor bounds checking of input data in web applications. Most sites that are vulnerable to this and have the potential for significant loss often use SSL for these transactions. This presents problems in detection since the IDS can not understand the encrypted data. In this style of attack it is possible for an attacker to acquire sensitive customer information such as credit card numbers, bank account numbers, user accounts, user passwords, change order entries and status and get sensitive corporate information or database schemas. In summary, the attacker can get any information that is in the database. A full description of how this attack works is available in this paper by Chris Anly http://www.nextgenss.com/papers/advanced_sql_injection.pdf and in this white paper by Kevin Spett http://cgisecurity.net/lib/SQLInjectionWhitePaper.pdf

Attack Mechanism:
This is a subtle interactive attack. Automated tools are not necessarily effective since there are many variables and much interpretation that is needed to successfully execute The attack.

Correlations:
No correlation of target system logs or firewall logs is possible in this attack due to the environment of this device.

Evidence of Active Targeting:
This is active targeting. The attacker must first browse the web sight and in this example appears to be attempting to take advantage a user sign-in or sign-up form.

Severity: 5

Criticality – **5** This is a E-commerce enabled web site with database access.

Lethality – **5** This attack could potentially be very damaging to a corporation and corporate image.

System Countermeasures –3 This system was a well patched modern system but could have potential application programming issues.

Network Countermeasures – 2 The firewall allowed the traffic but the IDS shunned the attacker thus removing the potential threat. There is no evidence that the attack was successful.

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

(5 +5) – (3 –2) = 5

Defensive Recommendations:

To properly defend against this attack the web server should have current security patches in place for the web server and all associated scripting languages the site uses. In addition to it is necessary to do proper checking of user inputted data to strip any possible characters that can allow for this attack to be successful. Web site source code audits and application vulnerability testing should be done. In H.D. Moore's presentation on SQL insertion he estimated that 50 % of all database driven sites for medium size companies are vulnerable. For smaller company's 75% are vulnerable. The document is located at http://www.digitaloffense.net/confs/bootcamp02/jpeg/sql/Slide01.html

A IDS will have a difficult time defending against this sort of attack. Some of the problems with detecting these attacks from an IDS viewpoint are encryption and the difficulty in writing signatures that will not false alarm often. To write effective signatures it is necessary for the IDS to have robust regular expression pattern matching capabilities. It is also necessary to have multiple signatures to detect this. As alluded to earlier the signature that detected this was less than perfect.

Original SQL insertion signature

```
RecordOfStringName   8010     80       1         1
".*=.*[Ss][Ee][Ll][Ee][Cc][Tt].*[Ff][Rr][Oo][Mm]"
```

Other signatures can be developed which may false alarm less. Suggested replacement signatures are listed below.

SQL insertion signatures

```
RecordOfStringName   8010     80       1         1
"[Gg][Ee][Tt].*=%27.*[Uu][Nn][Ii][Oo][Nn]"
```

```
RecordOfStringName   8011    80      1       1
"[Pp][Oo][Ss][Tt].*=%27.*[Uu][Nn][Ii][Oo][Nn]"
```

```
RecordOfStringName   8012    80      1       1       "[Gg][Ee][Tt].*=%27=1"
```

```
RecordOfStringName   8013    80      1       1       "[Gg][Ee][Tt].*=%27%3d1"
```

```
RecordOfStringName   8014    80      1       1       "[Pp][Oo][Ss][Tt].*=%27=1"
```

```
RecordOfStringName   8015    80      1       1       "[Pp][Oo][Ss][Tt].*=%27%3d1"
```

These first two signature varies from our original in that %27 (which is a hex encoded ‘) has been added and union replaces select and from qualifiers as well as making the request dependent on a get or post method. The advantages to this is that most of the insertion techniques use a hex encoded ‘ and union statement to add on to the normal SQL query. The last four signatures look for get and post methods involving a ‘=1 (which is a general return true statement). These signatures require the ‘ be hex encoded and the = could either be hex encoded as 3D or just used as an = sign.

Multiple Choice Test Question:

This is most likely an example of

GET /accounts/login.php?fname=john&lname=%27=1&password=%27%3D1 HTTP/1.0\r\n\r\n

A., A normal user login to a web site.
B., An attempt to use SQL insertion techniques to extract information from a database.
C., A user getting a normal web page by the above name.
D., Both A and C could be correct.

Answer – B This is most likely an attempt to use SQL injection techniques to get extract information from a database.

Detect #3: Syn-Fin FTP scan

This detect was a SYN-FIN scan directed towards a lab which included a Cisco Secure IDS, Windows 2000 Web Server protected by Entercept, and a Snort IDS. The attacker was scanning for ftp.

Cisco Secure IDS Logs

```
4,1012858,2002/04/17,00:32:55,2002/04/16,19:32:55,10008,100,101,OUT,IN,1,3000
,21,TCP/IP,143.248.62.88,W.X.Y.122,21,21,0.0.0.0,1638201219
4,1012859,2002/04/17,00:32:55,2002/04/16,19:32:55,10008,100,101,OUT,IN,3,3041
,0,TCP/IP,143.248.62.88,W.X.Y.122,21,21,0.0.0.0,
4,1012860,2002/04/17,00:32:55,2002/04/16,19:32:55,10008,100,101,OUT,IN,1,3000,
21,TCP/IP,143.248.62.88,W.X.Y.123,21,21,0.0.0.0,1638201219
4,1012861,2002/04/17,00:32:55,2002/04/16,19:32:55,10008,100,101,OUT,IN,1,3000,
21,TCP/IP,143.248.62.88,W.X.Y.124,21,21,0.0.0.0,1638201219
```

A full description of the Cisco log format is available in Appendix B. The source IP generated four distinct log entries on the Cisco device consisting of two unique signatures. The signatures triggered were 3000 which is a TCP connection and a 3041 which is a SYN-FIN scan. The 3000 signature triggered to 3 distinct IP addresses W.X.Y.122-124 while signature 3041 only triggered once.

Snort Alert Logs

```
04/16-19:32:59.950000  [**] [111:13:1] spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection
[**] {TCP} 143.248.62.88:21 -> W.X.Y.122:21
04/16-19:32:59.957563  [**] [100:1:1] spp_portscan: PORTSCAN DETECTED to port 21 from
143.248.62.88 (STEALTH) [**]
04/16-19:32:59.970000  [**] [111:13:1] spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection
[**] {TCP} 143.248.62.88:21 -> W.X.Y.123:21
04/16-19:32:59.990000  [**] [111:13:1] spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection
[**] {TCP} 143.248.62.88:21 -> W.X.Y.124:21
04/16-19:34:07.990598  [**] [100:2:1] spp_portscan: portscan status from 143.248.62.88: 3 connections
across 3 hosts: TCP(3), UDP(0) STEALTH [**]
04/16-19:37:08.074214  [**] [100:3:1] spp_portscan: End of portscan from 143.248.62.88: TOTAL
time(0s) hosts(3) TCP(3) UDP(0) STEALTH [**]
```

The Snort logs show three SYN-FIN alerts, which is contrary to the Cisco Secure IDS logs only having one alert for SYN-FIN connection. The Snort alerts are detected by the Snort portscan pre-processor and are not unique signatures. The snort host logs everything using tcpdump with the command tcpdump –w /dumpfiles/tcpdump-$date.dmp &. The tcpdump logs are broken down by host. Packets sent by the attacker are highlighted in red.

Tcpdump Logs: host W.X.Y.122

```
19:32:59.950000 143.248.62.88.ftp > W.X.Y.122.ftp: SF [tcp sum ok] 1638201219:1638201219(0) win
1028 (ttl 18, id 39426, len 40)
19:32:59.950000 W.X.Y.122.ftp > 143.248.62.88.ftp: R [tcp sum ok] 0:0(0) ack 1638201221 win 0 (DF)
(ttl 255, id 0, len 40)
```

Tcpdump Logs: host W.X.Y.123

```
19:32:59.970000 143.248.62.88.ftp > W.X.Y.123.ftp: SF [tcp sum ok] 1638201219:1638201219(0) win
1028 (ttl 18, id 39426, len 40)
19:32:59.970000 W.X.Y.123.ftp > 143.248.62.88.ftp: S [tcp sum ok] 2605978220:2605978220(0) ack
1638201220 win 24656 <mss 1460> (DF) (ttl 60, id 63603, len 44)
19:33:03.330000 W.X.Y.123.ftp > 143.248.62.88.ftp: S [tcp sum ok] 2605978220:2605978220(0) ack
1638201220 win 24656 <mss 1460> (DF) (ttl 60, id 63604, len 44)
```

```
19:33:10.080000 W.X.Y.123.ftp > 143.248.62.88.ftp: S [tcp sum ok] 2605978220:2605978220(0) ack
1638201220 win 24656 <mss 1460> (DF) (ttl 60, id 63605, len 44)
19:33:23.580000 W.X.Y.123.ftp > 143.248.62.88.ftp: S [tcp sum ok] 2605978220:2605978220(0) ack
1638201220 win 24656 <mss 1460> (DF) (ttl 60, id 63606, len 44)
19:33:50.580000 W.X.Y.123.ftp > 143.248.62.88.ftp: S [tcp sum ok] 2605978220:2605978220(0) ack
1638201220 win 24656 <mss 1460> (DF) (ttl 60, id 63607, len 44)
19:34:44.570000 W.X.Y.123.ftp > 143.248.62.88.ftp: S [tcp sum ok] 2605978220:2605978220(0) ack
1638201220 win 24656 <mss 1460> (DF) (ttl 60, id 63608, len 44)
19:35:44.570000 W.X.Y.123.ftp > 143.248.62.88.ftp: S [tcp sum ok] 2605978220:2605978220(0) ack
1638201220 win 24656 <mss 1460> (DF) (ttl 60, id 63609, len 44)
19:36:44.560000 W.X.Y.123.ftp > 143.248.62.88.ftp: R [tcp sum ok] 1:1(0) ack 1 win 24656 (DF) (ttl 60,
id 63610, len 40)
```

Tcpdump Logs: host W.X.Y.124

```
9:32:59.990000 143.248.62.88.ftp > W.X.Y.124.ftp: SF [tcp sum ok] 1638201219:1638201219(0) win
1028 (ttl 18, id 39426, len 40)
19:32:59.990000 W.X.Y.124.ftp > 143.248.62.88.ftp: S [tcp sum ok] 2349397918:2349397918(0) ack
1638201220 win 16616 <mss 1460> (DF) (ttl 128, id 48310, len 44)
19:33:02.920000 W.X.Y.124.ftp > 143.248.62.88.ftp: S [tcp sum ok] 2349397918:2349397918(0) ack
1638201220 win 16616 <mss 1460> (DF) (ttl 128, id 48311, len 44)
19:33:08.930000 W.X.Y.124.ftp > 143.248.62.88.ftp: S [tcp sum ok] 2349397918:2349397918(0) ack
1638201220 win 16616 <mss 1460> (DF) (ttl 128, id 48312, len 44)
```

The attacker sent a SYN-FIN packet to host W.X.Y.122 and host 122 responds with a reset. The attacker send a SYN-FIN packet to host W.X.Y.123 and host 123 which is a listening Solaris 2.8 host using tcp-wrappers gladly responds with 7 ACK packets before finally sending a reset. The ACK packets each have a unique incrementing IP ID, which precludes them from being re-transmissions. Host W.X.Y.124, which is a Windows 2000 server responds the same way.

Source of the attack: Network 4

Attack was generated by:
Snort 1.8.6 and Cisco Secure IDS 3.1

Probability the source address was spoofed:
Performing a trace route to the source of this attack reveals the host is 18 hops away. This is not consistent with the TTL of the attacker, which is 18. These are forged packets, so it is conceivable that the TTL is forged as well.

Description of the attack:
This attack is really very common. The attacker uses the same source and destination ports. This is supposed to be an attempt to bypass a packet filtering device by using a common port such as FTP, Web or DNS. In trying to bypass a filtering device the attacker would most likely be more successful source porting from port 80. The attacker also sets the SYN and FIN flags in the packet. This combination may be used to bypass a firewall that incorrectly checks for SYN packets by only seeing if the particular flag is set

by itself. This subtle relationship can be best described by creating tcpdump filters for the two methods.

- tcp[13]=2
- tcp[13] &0x02 !=0

The first filter only catches packets in which the SYN bit is the only bit set, so in essence a packet with SYN and FIN would not meet those requirements. The second filter will detect any packet that the SYN flag is set regardless of other flags. These are obviously crafted packets.

Crafted Packets:

```
19:32:59.950000 143.248.62.88.ftp > W.X.Y.122.ftp: SF [tcp sum ok] 1638201219:1638201219(0) win
1028 (ttl 18, id 39426, len 40)
19:32:59.970000 143.248.62.88.ftp > W.X.Y.123.ftp: SF [tcp sum ok] 1638201219:1638201219(0) win
1028 (ttl 18, id 39426, len 40)
9:32:59.990000 143.248.62.88.ftp > W.X.Y.124.ftp: SF [tcp sum ok] 1638201219:1638201219(0) win
1028 (ttl 18, id 39426, len 40)
```

The TTL is very low, the sequence number is the same for all packets, the SYN and FIN flags are set, the window size is small at 1028 and the IP ID remains the same for all packets at 39426.

Attack Mechanism:
This is an automated attack, most likely someone of script kiddie level that is either using a tool written by someone else or fairly new to writing tools themselves. A non kiddie level attacker with thorough knowledge of networking would never be so loud. This can trigger several IDS alarms easily and is not very stealthy where IDS is concerned.

Correlations:
I submitted a Dshield query but have not received a response.

Evidence of Active Targeting:
No real evidence of active targeting. This is most likely an automated and unattended scan.

Severity: 0

Criticality –3 Two of these are IDS sensors. They are in a lab and not critical except for occasional testing.

Lethality – 3 It was successful reconnaissance, nothing more.

System Countermeasures –5 These are well protected and up to date hosts. The Windows 2000 server was running Entercept in protect mode.

Network Countermeasures – 1 There were no network countermeasure to this activity.

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

(3 + 3) – (5 +1) = 0

Defensive Recommendations:
This attack is really pretty simple to protect against. Most modern firewalls will not allow this activity through. Likewise, any recent IDS can detect these attacks with ease. An IDS could shun or reset the attacker since there is no chance of a false alarm on these attacks.

Multiple Choice Test Question:
Question: Given the packet below, which is obviously forged, use passive fingerprinting techniques to determine which operating system most likely sent this packet ?

```
9:32:59.990000 143.248.62.88.ftp > W.X.Y.124.ftp: SF [tcp sum ok]
1638201219:1638201219(0) win 1028 (ttl 18, id 39426, len 40)
```

A., Windows
B., Linux
C., Solaris
D., None of the above

Answer – D. No reasonable assumption can be made of this packet. It does not fit in with any modern operating system fingerprint. The only assumption that can be made is that it was most likely generated from a platform that has many tools for crafting IP packets.

Assignment 3: Analyze This

Analyze This!

Overview:

Var-log consulting analyzed six days worth of IDS log data, using the alerts, scans and out of spec logs from March 18th thru March 23rd for GIAC University. Following is a thorough analysis of this data. For the purpose of analysis all MY.NET addresses were converted to the 10.10 Network. A full description of the methodology used in distilling the data is listed in Appendix D.

Executive Summary:

There is significant evidence of active targeting directed towards GIAC University. Furthermore, from the log data provided there is evidence of compromised internal hosts and malicious activity being initiated from inside GIAC University. . Contained in this analysis are summary tables, threat graphs, link graphs, host, and event analysis. Events were chosen for analysis based on several key factors. These factors include

- Events with a high probability of being an external threat
- Scans that varied from a baseline, showing significant change from day to day
- Possible hosts that are compromised

A general recommendations section is contained at the end, which gives detailed recommendations that can help improve the overall security posture of GIAC University without adversely impacting the traditionally open University security policy.

Table 1: Event Totals

Event Type	Amount
Alerts	1,554,265
Scans	3,312,172
Out Of Spec	673

Alert Summary: Detects By Occurrence

The following tables contain a summary of the 20 most common events. Port Scans and Watch List alerts have been excluded. Port Scans are handled in the scans section.

Table 2: Alert Summary (Detects by Occurrence)

Alert Type	Total
Connect to 515 from inside	164,090
spp_http_decode: IIS Unicode attack detected	90307

IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize	74277
SMB Name Wildcard	73702
SNMP public access	39759
ICMP Echo Request L3retriever Ping	36585
MISC Large UDP Packet	26880
INFO MSN IM Chat data	16792
INFO Inbound GNUTella Connect request	12464
High port 65535 udp - possible Red Worm – traffic	11008
spp_http_decode: CGI Null Byte attack detected	10557
ICMP Echo Request Nmap or HPING2	6473
WEB-MISC Attempt to execute cmd	3702
FTP DoS ftpd globbing	3038
ICMP Fragment Reassembly Time Exceeded	2382
INFO Outbound GNUTella Connect request	1961
Possible trojan server activity	1911
SCAN Proxy attempt	1605
ICMP Router Selection	1498
WEB-IIS view source via translate header	992

The following two tables list the most common source and destination IP's and the alert most often associated with the particular IP and direction. IP's colored red are listed in both charts. An asterisk next to an IP denotes that there is enough suspicious traffic to warrant analysis of the traffic to the host. These IP's were chosen because log analysis of the traffic determined these IP's were either common targets or common sources of events that may be high threat.

Table 3: Alert Source Summary (Alert Top Talkers Source)

Source IP	Alerts	Most Common Alert
10.10.88.190*	74277	IDS552/web-iis_IIS ISAPI Overflow ida
10.10.70.177	20644	SNMP public access
10.10.11.6	16685	spp_portscan
10.10.11.7	13868	SMB Name Wildcard
10.10.153.119	11722	connect to 515 from inside
10.10.153.118	9524	connect to 515 from inside
10.10.153.124	8389	connect to 515 from inside
10.10.153.106*	7839	IIS Unicode attack detected
10.10.153.171	7645	connect to 515 from inside
10.10.153.211*	7600	IIS Unicode attack detected

Table 4: Alert Destination Summary (Alert Top Talkers Destination)

Destination IP	Alerts	Most Common Alert
10.10.150.198	163864	connect to 515 from inside
10.10.11.6	36147	ICMP Echo Request L3retriever Ping
10.10.11.7	30147	ICMP Echo Request L3retriever Ping
211.115.213.202	10709	IIS Unicode attack detected
10.10.11.5	10606	ICMP Echo Request L3retriever Ping
209.10.239.135	9449	CGI Null Byte attack
10.10.153.153*	7031	MISC Large UDP Packet
10.10.153.197*	6561	MISC Large UDP Packet
10.10.150.195	6152	SNMP public access
10.10.153.208	5854	INFO Inbound GNUTella Connect request

The following two tables list the most common source and destination IP's from external networks and the alert most often associated with the particular IP. IP's colored red are listed in both charts. An asterisk next to an IP denotes that there is enough suspicious traffic to warrant analysis of the traffic to the host. Several hosts are analyzed in the host analysis section.

Table 5: Alert Source Summary External (External Alert Top Talkers Source)

Source IP	Alerts	Address Space Owner	Most Common Alert
212.179.35.118	5148	ISDN Net Ltd (ISREAL)	Watchlist 000220 IL-ISDNNET-990517
208.191.18.173*	4934	American Association Of Petroleum	IIS Unicode attack detected
63.240.15.199	4429	AT&T CERFnet	MISC Large UDP Packet
63.240.15.204	3776	AT&T CERFnet	High port 65535 udp - possible Red Worm - traffic
63.240.15.207	3044	AT&T CERFnet	MISC Large UDP Packet
63.240.15.205	2714	AT&T CERFnet	MISC Large UDP Packet
202.98.15.138	2386	CC-MULTI-MEDIA-NET (CHINA)	MISC Large UDP Packet
212.179.27.176	1404	ISDN Net Ltd (ISREAL)	Watchlist 000220 IL-ISDNNET-990517
80.13.214.233*	1312	Wanadoo Interactive (FRANCE)	IIS Unicode attack detected
61.132.208.63	1253	CHINANET-AH (CHINA)	INFO - Possible Squid Scan

Table 6: Alert Destination Summary External (External Alert Top Talkers Destination)

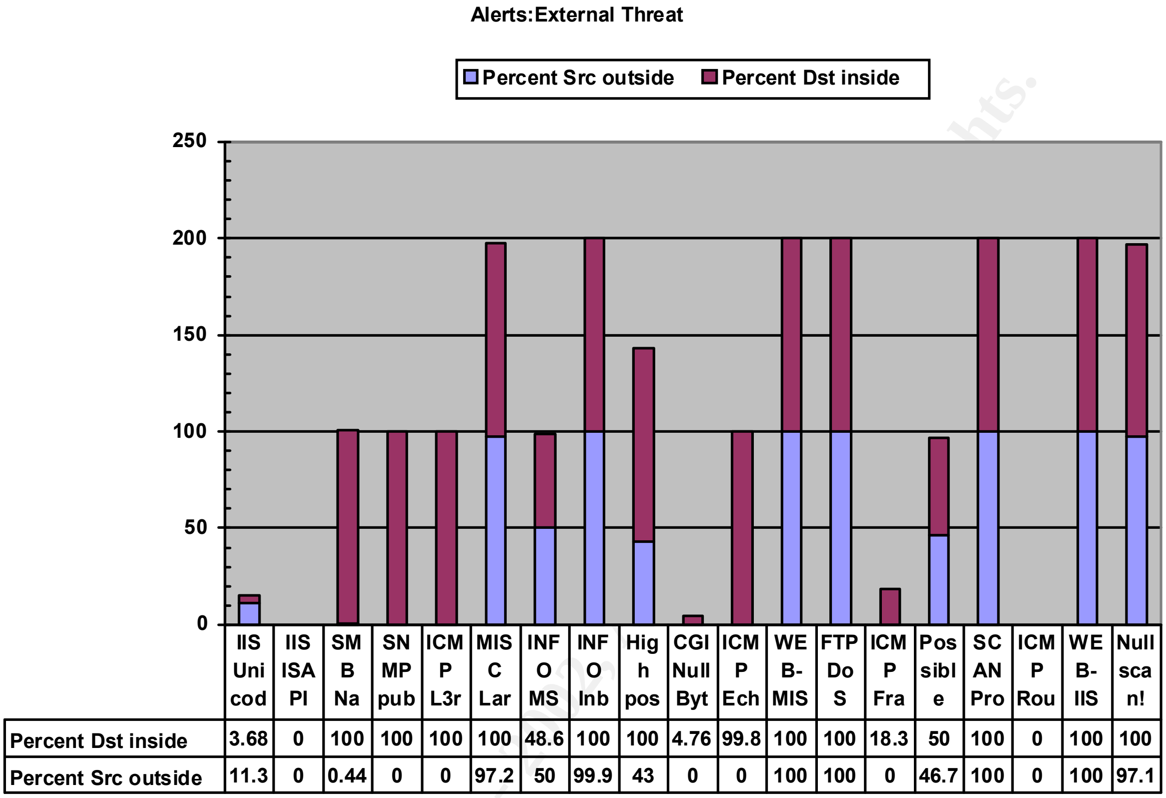
Destination IP	Alerts	Address Space Owner	Most Common Alert
211.115.213.202	10709	GCG-IDC (Korea)	IIS Unicode attack
209.10.239.135	9449	Globix Corporation	CGI Null Byte attack
211.32.117.26	2889	DACOM (Korea)	IIS Unicode attack
211.115.213.207	2517	GCG-IDC (Korea)	IIS Unicode attack
64.12.184.141	2128	AOL	IIS Unicode attack
211.233.85.9	1577	KIDC (Korea)	IIS Unicode attack
211.233.29.215	1523	KIDC (Korea)	IIS Unicode attack
211.233.85.62	1514	KIDC (Korea)	IIS Unicode attack
224.0.0.2	1498	MCAST	ICMP Router Selection
211.233.29.207	1444	KIDC (Korea)	IIS Unicode attack

Threat Source Summary:
The following graph depicts the alerts and their associated external threat. This was obtained by taking a total count for each alert, then counting the number of unique addresses that triggered each alert. Once a total count of the unique addresses responsible for each alert was obtained, the addresses were broken down by the source and destination networks. Alerts of a source network that was not 10.10 (MY.NET) or a destination that was 10.10 (MY.NET) were considered to be external threats. The following table will help explain this graph.

Table 7: Graph Explanation

Alert Destination	Alert Source	Threat Level
Destination inside	Source outside	Very High (purple & blue)
Destination inside	Source inside	Medium (purple)
Destination outside	Source inside	Low
Destination outside	Source outside	Not shown (Should not be seen)

Graph 1: Threat Source Summary (top 20 alerts)



	IIS Uni cod	IIS ISA PI	SM B Na	SN MP pub	ICM P L3r	MIS C Lar	INF O MS	INF O Inb	Hig h pos	CGI Null Byt	ICM P Ech	WE B- MIS	FTP Do S	ICM P Fra	Pos sibl e	SC AN Pro	ICM P Rou	WE B- IIS	Null sca n!
Percent Dst inside	3.68	0	100	100	100	100	48.6	100	100	4.76	99.8	100	100	18.3	50	100	0	100	100
Percent Src outside	11.3	0	0.44	0	0	97.2	50	99.9	43	0	0	100	100	0	46.7	100	0	100	97.1

The following table summarizes the threats that are more likely from external sources. Potential impact is associated with each of these alerts. Alerts of high impact will be analyzed in the event analysis section. The impact table describes the possible types of activity associated with each impact rating.

Table 8: Impact Ratings

Impact Level	Impact
High	Potential Compromise
	Potential Denial Of Service
Medium	Recognisance and Enumeration
Low	Misuse of resources

Table 9: Most Common Alerts, External

Alert	Count	Impact
MISC Large UDP Packet	26880	High
INFO Inbound GNUTella Connect request	12464	Low
High port 65535 udp - possible Red Worm – traffic	11008	High

WEB-MISC Attempt to execute cmd	3702	High
FTP DoS ftpd globbing	3038	High
SCAN Proxy attempt	1605	Medium
WEB-IIS view source via translate header	992	Medium
Null Scan	839	Medium

Conversely, by looking at the graph in reverse we can assume that alerts that are at a very low level on the graph are sourced from inside of GIAC Universities network and destined outside. The following table depicts these alerts with their associated impact. This may signify compromised hosts or someone from within the network attempting to compromise external hosts.

Table 10: Most common Alerts, Internal

Alert	Count	Impact
IIS Unicode attack	90307	High
IIS ISAPI Overflow ida	74277	High
CGI Null Byte attack	10557	Medium
ICMP Fragment Reassembly Time Exceeded	2382	Medium
ICMP Router Selection	1498	Low

Scan Summary:
Port scans were distilled using the same methods as the alerts were. Following is summary of the most common source port and destination ports. Ports common to both tables are highlighted in red.

Table 11:Scans Most Common Destination Port

Port	Amount	Service
7001	439211	Chat
80	437454	HTTP
7000	256844	Chat
1346	215415	Alta Analytics
53	191773	DNS
0	132627	Not used
4665	103764	EDonkey
137	89347	Netbios-ns
514	83794	Syslog
6346	55625	Gnutella

Table 12: Scans Most Common Source Port

Port	Count	Service
7000	439100	Chat
123	386018	NTP

7001	293629	Chat
1347	215379	bbn-mmc
0	162729	Not used
137	89629	Netbios-ns
514	73075	Syslog
1257	51023	Shockwave2
88	50560	Kerberos
1753	32943	Translogic-lm

Table 13 represents the internal IP addresses that were most often scanned.

Table 13: Most Commonly scanned Destination IP Internal

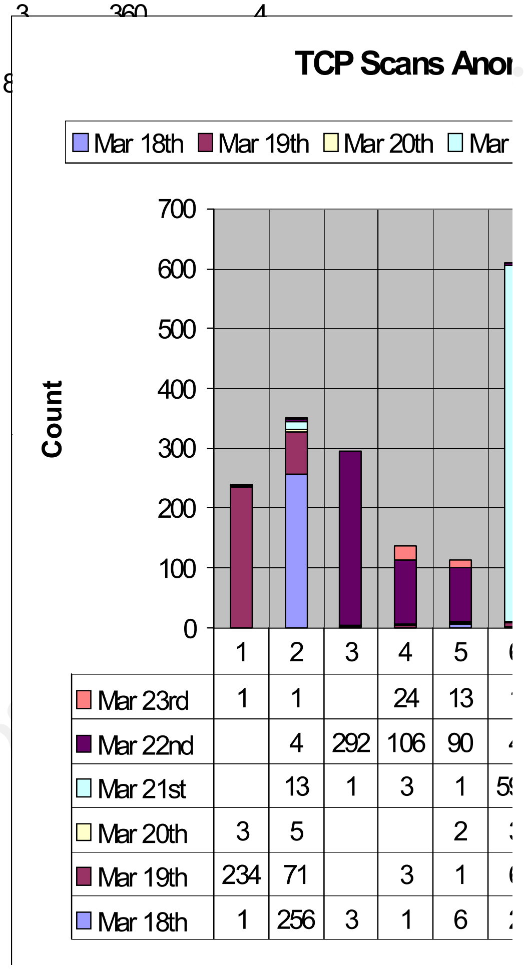
Count	IP	Port
109719	10.10.1.3	53
83681	10.10.1.7	514
77365	10.10.6.45	7000
73520	10.10.1.4	53
49532	10.10.60.43	7000
28503	10.10.5.55	137
26241	10.10.6.53	7000
25707	10.10.5.50	137
18029	10.10.6.49	7000
16002	10.10.6.53	7000

Malicious scans:

While scans are useful for identifying traffic patterns the most common patterns are often not malicious in nature and are the result of normal traffic. The following graphs depict scans in which the percentage of activity destined to a certain port varied tremendously from day to day. These graphs represent scans destined to the internal network and mathematically work by detecting a certain degree of change from average. To remove changes in traffic from day to day every port/count combination was converted to a percentage of that day's activity. Then all days for that port combination were summed together and divided by the amount of days data used (which was 6). Then ports, which exceeded a certain multiplier of their average for any day, were represented in the graph. This multiplier was somewhat arbitrary, and was manipulated until eight to twelve ports became apparent. This limit was really imposed by the limitations of readable functional graphs. In the case of UDP scans a certain minimum qualifying percentage was used to help further distill the data. The scripts used to generate this udp_report.pl and tcp_report.pl are included in Appendix C.

Link Graph 1: TCP Scans Anomalous (Link Graph #1)

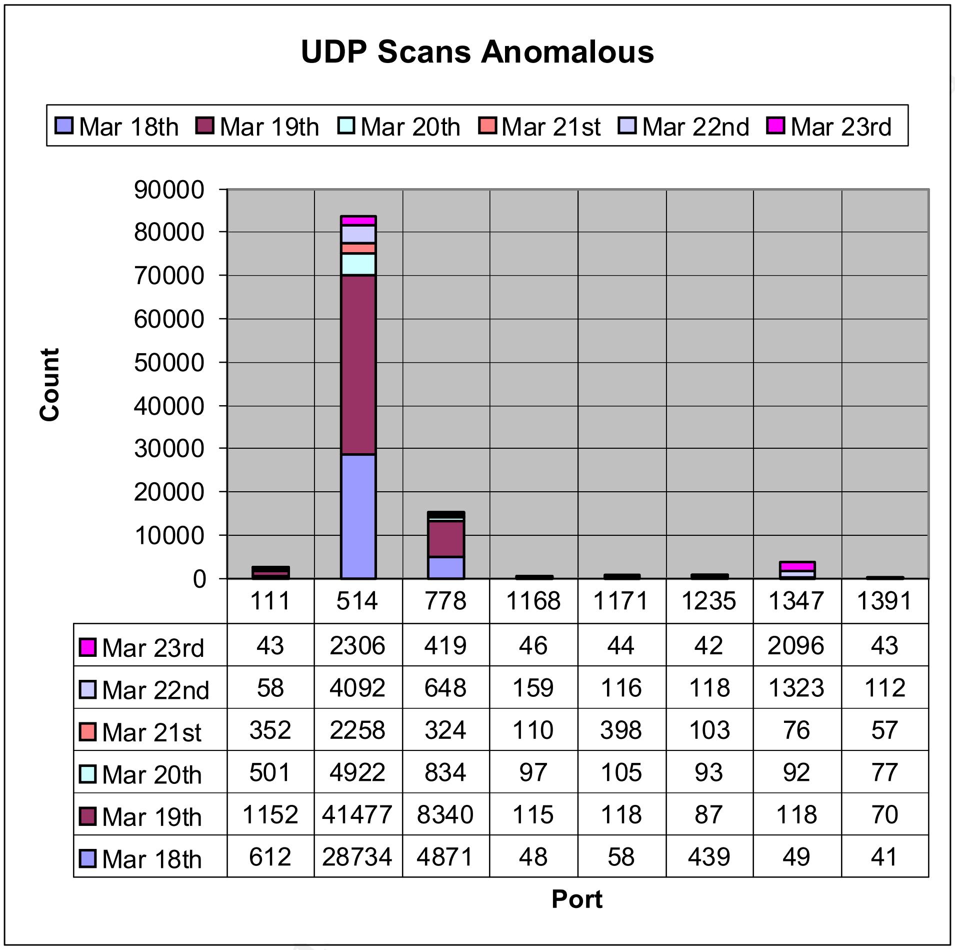
8080	5	5	3	360	4
8888	1				
12345	9		8		
13000					



	1	2	3	4	5
🟥 Mar 23rd	1	1		24	13
■ Mar 22nd		4	292	106	90
□ Mar 21st		13	1	3	1
□ Mar 20th	3	5			2
■ Mar 19th	234	71		3	1
□ Mar 18th	1	256	3	1	6

From the above link graph a few combination stand out. Port 0 was scanned excessively on Mar 19th, Port 22 on Mar 18th, the proxy port combination of 1080, 3128, 8000 and 8080 on the 21st and 8888 and 99 on 22nd. The last two combinations will be examined in greater detail later in the event analysis section.

Link Graph 2: UDP Scans Anomalous (Link Graph #2)



	111	514	778	1168	1171	1235	1347	1391
Mar 23rd	43	2306	419	46	44	42	2096	43
Mar 22nd	58	4092	648	159	116	118	1323	112
Mar 21st	352	2258	324	110	398	103	76	57
Mar 20th	501	4922	834	97	105	93	92	77
Mar 19th	1152	41477	8340	115	118	87	118	70
Mar 18th	612	28734	4871	48	58	439	49	41

From the above link graph we can see that scans destined to port 514 and 778 were very high on the 18th and 19th. Conversely scans to port 1347 were very high on the 22nd and 23rd.

Out Of Spec Packet Summary:

There were 673 out of spec packet alerts for the period from March 18th thru March 23rd. Of these events, 611 were generated by two hosts 64.152.183.174 and 217.56.233.186. Another 15 of these events were generated by 210.83.45.86. The events from these three hosts will be addressed in the event analysis section.

Internal host analysis:

Due to the nature of the alarms these hosts either received or generated there is sufficient cause to further analyze these specific hosts. I will be using the standard rules provided with Snort-1.8.4 for this analysis.

10.10.88.190
10.10.153.211

10.10.153.106
10.10.153.153
10.10.153.197

Host: 10.10.88.190
This host was the source of 74277 “IIS ISAPI Overflow “alarms. This most relevant Snort signature for this alarm triggers on any .ida attempt with a payload size greater than 240 bytes, which is associated with the .ida buffer overflow.

Snort Signature: ida

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS ISAPI
.ida attempt"; uricontent:".ida?"; nocase; dsize:>239; flags:A+;
reference:arachnids,552; classtype:web-application-attack; reference:cve,CAN-
2000-0071; sid:1243; rev:2;)
```

Most of these alerts trigger on the 18th of March. This type of activity is most closely associated with the Code Red or Nimda (Concept) worms. Since this host did not trigger other alarms that are associated with Nimda it is believed that this host was infected with Code Red. For more information on Code Red a full analysis by eEye Digital Security is available at http://www.eeye.com/html/Research/Advisories/AL20010717.html . Do to the nature of this particular infection and how the worm chose it’s victims it is believed that this is Code Red version 1. The reason for this assumption is that Code Red 2 scanned more within the local class A and B networks, thus the destination IP’s being attacked from this host would have attacked more often within GIAC Universities network. This conclusion is based on the fact that this worm appears to not be heavily favoring any particular networks. This host appears to have only scanned until early on March 19th so I believe the initial infection problem has been resolved. To prevent this attack from being successful the system must be patched or reconfigured. The Microsoft patch is available at http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp

Host: 10.10.153.211
Host 10.10.153.211 was the source of 7,600 “Unicode attack detected” alerts and 11,611 alerts overall. Most of the Unicode alerts were on Mar 18th and Mar 21st.

Table 14: Alerts associated with 10.10.153.211

Count	Alert	Source of Destination
7041	IIS Unicode Attack	Source
3713	Spp_portscan	Source
524	INFO MSN IM Chat	Both
294	Connect to 515	Source
30	Possible Red Worm	Destination

5	Scan Proxy attempt	Destination
2	ICMP Echo NMAP or HPING	Destination
1	SYN-FIN	Destination
1	INFO possible Squid scan	Destination

Due to this host using MSN Instant Messenger and the fact that the Unicode attacks were inconsistent this appears to a malicious user rather than a compromised host.

Host: 10.10.153.106
This host was associated with a total of 11,038 alerts of which 7,839 were Unicode attacks. This host appears to be targeting the same networks with Unicode attacks as 10.10.153.211. Once again this appears to be active targeting by a malicious user. A breakdown of the alerts associated with this host reveals striking similarity to 10.10.153.211 which is shown in table 14. This host however, did generate 500 ICMP Fragment Reassembly Time Exceeded alarms on Mar 21st. The alarms are in response to the host receiving the first fragments of a series of fragmented packets but not receiving any others. After a Operating System specific amount of time the receiving host then sends back an ICMP Fragment Reassembly time exceeded message. The true source of these alerts is 212.73.235.74. This IP (212.73.235.74) generated this alert to another internal host 10.10.153.167 as well.

Host: 10.10.153.153
The internal host 10.10.153.153 generated 16,341 total alerts of which 4053 were portscans directed at various ports. Another 2816 of these alerts were CGI Null Byte attacks directed at 209.10.239.135. Another 2176 alerts for Unicode attacks were generated. Most of these were directed at the 211.233 network, which is in Korea. These attacks were spread out over several days. A total breakdown of all alerts for this host was as follows.

Table 15: Alerts associated with 10.10.153.153

Count	**Alert**	**Source of Destination**
6799	Large UDP Packet	Destination
4053	Portscan	Source
2816	CGI Null Byte	Source
2176	Unicode	Source
185	Possible Red Worm	Destination
160	Fragmentation Time Exceeded	Destination
93	Connect to 515	Source
21	Scan Proxy Attempt	Destination
11	Exploit x86 NOOP	Destination
11	Possible Squid scan	Destination
9	Possible IRC access	Source

4	EXPLOT NTPDX buffer overflow	Destination
2	ICMP NMAP or HPING	Destination
1	SYN-FIN scan	Destination

In addition to the scanning this host has been involved in it appears that this host may have been the target of a NTPDX buffer overflow attempt on March 20th and 21st as well as may have been the target of several UDP based attacks on the 18th and 19th. These attacks will be covered in more detail in the event analysis section. This host actively participated in several attacks mostly directed towards Korean Internet address space.

Host: 10.10.153.197
This host was either the source or destination in 14,524 alerts of which 9.957 were portscans. Many of these scans originate from port 6970 and use UDP as the protocol. This port and protocol combination is associated with RTP so it is possible many of these scans may be normal user traffic. This host initiated 483 Unicode alerts directed mainly towards 211 and 203 address space which are Korean. Alerts associated with this host were similar to 10.10.153.153. This actively participated in attacks directed towards Korean networks.

Event Analysis:
This section contains analysis of alerts and scans chosen through the distillation process. For this section each event will follow a standard format.

Event: WEB-MISC Attempt to execute cmd and IIS Unicode

This attack was chosen because it is viewed as a substantial external threat from graph 1. There were 3702 alerts generated, of which the source of the attack was 100% outside of GIAC Universities network and the destination was always a host inside GIAC Universities network.

Source of the attack:
GIAC Universities network. 27 different external hosts generated this alert. 2450 alerts were generated from a single host 208.191.18.173. Discussion of this attack will focus primarily on this host.

Attack was generated by:
Snort IDS. The most likely signature to trigger this from Snort 1.8.4 signature base is the signature listed below. This signature may not be an exact match as the message is different. This signature looks for the URI content “cmd?&”.

Signature: cmd

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS cmd?
acess";flags: A+; content:".cmd?&"; nocase; classtype:web-application-attack;
sid:1003; rev:2;)
```

Probability the source address was spoofed:
There is very little probability the source address was spoofed. The attack requires a full TCP connection. Running an nslookup on this IP returns nothing. A whois query using geektools (http://geektools.com) shows the owner as American Association of Petroleum as the owner of the address space from 208.191.18.168 208.191.18.175.

Whois: 208.191.18.173

```
Southwestern Bell Internet Services (NETBLK-SBIS2) SBIS2
                                  208.188.0.0 - 208.191.255.255
American Association of Petroleum (NETBLK-SBCIS81285) SBCIS81285
                                 208.191.18.168 - 208.191.18.175
```

Doing a visual trace route using http://visualroute.backland.net shows the owner of the IP to be in Okemah Oklahoma.

Description of the attack:
This attack took place on March 20th and lasted just over 5 hours. The attack was against six internal hosts

Host Summary

Host	Count
10.10.150.246	1418
10.10.150.220	777
10.10.150.41	691
10.10.150.143	658
10.10.150.59	612
10.10.150.101	397
10.10.88.217	381

Attack Mechanism:
This attack attempts to execute commands outside the web server's root directory. The attacker usually tries to execute cmd.exe. If the attacker is successful the attacker will be able to access cmd.exe with the privilege of the web server which is normal user privileges. This particular attack can actually be a part of several other attacks. Attacks usually associated with this can be Unicode directory traversal, Code Red and the Concept Worm. Attackers commonly will deface the web site, perform denial of service or possibly execute commands to upload programs to the server for further access. This particular attack generated two other alerts other than Unicode or cmd.exe.

Alerts

```
spp_http_decode: IIS Unicode attack detected [**] 208.191.18.173:1741 ->
```

```
10.10.150.41:80
WEB-MISC 403 Forbidden [**] 10.10.150.41:80 -> 208.191.18.173:1742
WEB-IIS Unauthorized IP Access Attempt [**] 10.10.150.41:80 -> 208.191.18.173:1732
WEB-MISC Attempt to execute cmd [**] 208.191.18.173:1740 -> 10.10.150.41:80
```

Since this attack only generated these alerts and did not generate ida overflow, which would associate the attack with Code Red, nor did it generate a root.exe alerts which would associate the attack with the Concept worm it is believed this was some other variant, or a possible directed attack. In his practical, Guofei Jiang gives a full description of the mechanics of the Unicode attack at http://www.sans.org/newlook/digests/unicode.htm.

Correlations:
There was no other activity from this host

Defensive Recommendations:
Microsoft has recently released a cumulative patch that repairs this as well as several other recent IIS vulnerabilities. The patch is available at http://www.microsoft.com/security/security_bulletins/ms02018_iis.asp.
Microsoft also has an IIS lockdown tool available at http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/tools/locktool.asp. Most security breaches can be avoided by staying current on system security updates.

Event: NTPDX buffer overflow

This event was chosen because several inside hosts were attacked using this particular attack. This is a particularly dangerous attack that can yield administrator access.

Source of the attack:
Several inside hosts were attacked between March 18th and March 22nd from six distinct sources.

Logs

```
03/18-20:18:59.768424  [**] EXPLOIT NTPDX buffer overflow [**] 213.248.114.96:256 ->
10.10.153.152:123
03/19-20:06:32.445546  [**] EXPLOIT NTPDX buffer overflow [**] 63.146.181.119:123 ->
10.10.152.246:123
03/19-20:06:33.098635  [**] EXPLOIT NTPDX buffer overflow [**] 63.146.181.119:123 ->
10.10.152.246:123
03/19-20:06:33.742495  [**] EXPLOIT NTPDX buffer overflow [**] 63.146.181.119:123 ->
10.10.152.246:123
03/19-20:06:34.398952  [**] EXPLOIT NTPDX buffer overflow [**] 63.146.181.119:123 ->
```

```
10.10.152.246:123
03/20-22:56:14.155424  [**] EXPLOIT NTPDX buffer overflow [**] 211.233.25.32:1354 ->
10.10.153.153:123
03/20-22:56:14.897141  [**] EXPLOIT NTPDX buffer overflow [**] 211.233.25.32:1354 ->
10.10.153.153:123
03/21-20:12:57.345314  [**] EXPLOIT NTPDX buffer overflow [**] 63.250.205.9:1271 ->
10.10.153.185:123
03/21-21:43:38.106908  [**] EXPLOIT NTPDX buffer overflow [**] 63.250.205.44:1239 ->
10.10.153.153:123
03/21-21:43:38.447766  [**] EXPLOIT NTPDX buffer overflow [**] 63.250.205.44:1239 ->
10.10.153.153:123
03/22-08:56:31.973364  [**] EXPLOIT NTPDX buffer overflow [**] 66.38.171.141:18100 ->
10.10.150.215:123
```

Attack was generated by:

Signature: NTPDX

```
alert udp $EXTERNAL_NET any -> $HOME_NET 123 (msg:"EXPLOIT ntpdx
overflow attempt"; dsize: >128; reference:arachnids,492; classtype:attempted-
admin; sid:312; rev:1;)
```

Probability the source address was spoofed:

Very little, this attack does use UDP, which can easily be spoofed. The attacker, however is trying to get shell access. The following is a whois on the the attacking hosts.

Attacker 1: 213.248.114.96

```
Query:     213.248.114.96
Registry:  whois.ripe.net
Results:
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:     213.248.114.64 - 213.248.114.127
netname:     JOINTMODERN
descr:       Joint Modern Ltd
descr:       Only for Co-location
descr:       London
country:     GB
admin-c:     CM251-RIPE
tech-c:      AR346-RIPE
status:      ASSIGNED PA
mnt-by:      TELIANET-LIR
```

```
notify:      mntripe@telia.net
changed:     jessica@telia.net 20011128
source:      RIPE

route:       213.248.64.0/18
descr:       TELIANET-BLK
remarks:     Abuse issues should be reported at
remarks:     http://www.telia.com/security/
remarks:     Mail to abuse@telia.net will be auto-replied
remarks:     and referred to the URL above.
origin:      AS1299
mnt-by:      TELIANET-RR
changed:     rr@telia.net 20010514
source:      RIPE

person:      Charles Moss
address:     Hudson House
address:     Hudson Way
address:     Derby
e-mail:      charles.moss@mediawave.co.uk
phone:       +44-1332-866700
fax-no:      +46-1332-208485
nic-hdl:     CM251-RIPE
notify:      mntripe@telia.net
changed:     jessica@telia.net 20011128
source:      RIPE

person:      Andy Ringer
address:     Hudson House
address:     Hudson Way
address:     Derby
e-mail:      andy.ringer@mediawave.co.uk
phone:       +46-1332-866700
fax-no:      +46-1332-208485
nic-hdl:     AR346-RIPE
notify:      mntripe@telia.net
changed:     jessica@telia.net 20011128
source:      RIPE
```

Attacker 2: 63.146.181.119

```
Query:    63.146.181.119
Registry: whois.arin.net
Results:
Qwest Communications (NETBLK-NET-QWEST-BLKS-2) NET-QWEST-BLKS-2
                              63.144.0.0 - 63.151.255.255
```

```
Scale 8 (NETBLK-QWEST-IAD-SCALE81) QWEST-IAD-SCALE81
                                   63.146.180.0 - 63.146.181.255
```

Attacker 3: 211.233.25.32

```
P Address        : 211.233.25.0-211.233.25.63
Network Name        : KIDC-INFRA-COLOCATION
Connect ISP Name   : KIDC
Connect Date        : 20001201
Registration Date  : 20001220

[ Organization Information ]
Orgnization ID     : ORG141241
Org Name           : KIDC
State            : SEOUL
Address           : 261-1 Nonhyun-dong Kangnam-ku
Zip Code           : 135-010

[ Admin Contact Information]
Name              : SangGyu Jang
Org Name           : KIDC
State            : SEOUL
Address           : 261-1 Nonhyun-dong Kangnam-ku
Zip Code           : 135-010
Phone             : +82-2-6440-2920
Fax              : +82-2-6440-2909
E-Mail            : support@kidc.net

[ Technical Contact Information ]
Name              : TaeUng Kim
Org Name           : KIDC
State            : SEOUL
Address           : 261-1 Nonhyun-dong Kangnam-ku
Zip Code           : 135-010
Phone             : +82-2-6440-1965
Fax              : +82-2-6440-2909
E-Mail            : ip@kidc.net
```

Attacker 4: 63.250.205.9 & 63.250.205.44

```
Query:     63.250.205.9
Registry:  whois.arin.net
Results:
Yahoo! Broadcast Services, Inc. (NETBLK-NETBLK2-YAHOOBS)
   701 First Avenue
   Sunnyvale, California 94089
```

```
 US

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

 Coordinator:
    Admin, Netblock  (NA258-ARIN)  netblockadmin@yahoo-inc.com
    1-408-349-5555

 Domain System inverse mapping provided by:

 NS1.YAHOO.COM             66.218.71.63
 NS2.YAHOO.COM             209.132.1.28
 NS3.YAHOO.COM             217.12.4.104
 NS4.YAHOO.COM             63.250.206.138
 NS5.YAHOO.COM             64.58.77.85

 ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

 Record last updated on 27-Mar-2002.
 Database last updated on  19-Apr-2002 19:58:48 EDT.
```

Attacker 5: 66.38.171.141

```
Query:    66.38.171.141
Registry:  whois.arin.net
Results:
GT Group Telecom Services Corp. (NETBLK-GROUPTELECOM-BLK-3)
GROUPTELECOM-BLK-3
                                  66.38.128.0 - 66.38.255.255
Streaming Media Corp. (NETBLK-GT-66-38-171-0) GT-66-38-171-0
                                  66.38.171.0 - 66.38.171.255
```

Description of the attack:
There were six different attackers trying to exploit five different internal hosts. Each one of these external hosts performed several scans of the network and had other triggered other alarms including possible Red Worm traffic.

Logs (Brief)

```
spp_portscan: portscan status from 63.146.181.119: 7 connections across 1 hosts:
TCP(0), UDP(7) [**]
pp_portscan: portscan status from 213.248.114.96: 7 connections across 1 hosts: TCP(0)
, UDP(7) [**]
```

```
pp_portscan: portscan status from 211.233.25.32: 7 connections across 1 hosts: TCP(0),
UDP(7) [**]
spp_portscan: PORTSCAN DETECTED from 63.250.205.9 (THRESHOLD 4
connections exceeded in 4 seconds) [**]
spp_portscan: portscan status from 66.38.171.141: 1 connections across 1 hosts: TCP(0),
UDP(1) [**]
```

Attack Mechanism:
This attack targets a buffer overflow in the Network Time Protocol daemon. An attack description with buffer overflow code is available at http://online.securityfocus.com/archive/1/174011 The Cert advisory is available at http://www.kb.cert.org/vuls/id/970472

Correlations:
There are correlations to other attacks upon the same network which will be analyzed next.

Defensive Recommendations:
This vulnerability has been known since April 2001 and affects many Unix / Linux variants. Patches are available.
FreeBSD
ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/i386/packages-3-stable/net/ntp-4.0.99k_2.tgz
RedHat Linux
`ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/i386/packages-3-stable/net/ntp-4.0.99k_2.tgz`

Event : High port 65535 udp - possible Red Worm – traffic

This alert was chosen for analysis because of the significantly large amount of alerts from this signature and because the threat was determined from graph 1 to be mostly external.

Source of the attack:
This alert triggered 11008 times from 203 distinct source addresses. Since hosts that triggered the NPDX alerts also triggered these the focus will be on those hosts.

Count	Source IP
57	213.248.114.96
18	66.38.171.141
12	63.250.205.44
10	63.250.205.9
7	63.146.181.119
3	211.233.25.32

Attack was generated by:

The current Snort 1.8.4 rule set does not have a specific rule for the Red Worm / Adore worm. The signature appears to alert on any TCP or UDP traffic that involves port 65535.

Probability the source address was spoofed:
Very little, all of the traffic uses UDP which is easily spoofed but due to the variety of alerts generated and the fact the attacker is attempting to execute a buffer overflow spoofing is unlikely. I can not verify this by doing a trace back since I do not have access to GIAC university's network.

Description of the attack:
All of the IP's share common attacks. The alerts have a similar format to below.

```
19-20:06:34.398952  [**] EXPLOIT NTPDX buffer overflow [**] 63.146.181.119:123 -
> 10.10.152.246:123
03/19-20:19:39.475393  [**] spp_portscan: portscan status from 63.146.181.119: 4
connections across 1 hosts: TCP(0), UDP(4) [**]
03/19-20:19:41.956346  [**] spp_portscan: portscan status from 63.146.181.119: 3
connections across 1 hosts: TCP(0), UDP(3) [**]
03/19-20:00:43.046064  [**] High port 65535 udp - possible Red Worm - traffic [**]
63.146.181.119:65535 -> 10.10.152.246:65280
```

Attack Mechanism:
The "Red Worm" or Adore worm as it is otherwise known scans the Internet looking for Linux hosts that are vulnerable to rpc-statd, wu-ftpd, LPRng and BIND. Once a host is compromised the worm changes configuration files, attempts to send system specific information via email and replaces the ps binary with a trojaned version. The NTPDX vulnerability is not associated with the Red Worm. Examining the scan files, several of these hosts have very similar traffic patterns. These patterns all contain the following UDP activity as well as random high port to high port activity.
Source port 0 -> Destination port 0
Source port 516 -> Destination port 1588
Source port 7000 -> Destination port 7001

Host: 213.248.114.96

```
Mar 18 19:40:58 213.248.114.96:516 -> 10.10.153.152:1588 UDP
Mar 18 19:40:59 213.248.114.96:30511 -> 10.10.153.152:25719 UDP
Mar 18 19:41:02 213.248.114.96:0 -> 10.10.153.152:0 UDP
Mar 18 19:41:00 213.248.114.96:55278 -> 10.10.153.152:1407 UDP
Mar 18 19:41:03 213.248.114.96:7000 -> 10.10.153.152:7001 UDP
```

Host: 63.146.181.119

```
Mar 19 19:51:40 63.146.181.119:516 -> 10.10.152.246:1588 UDP
Mar 19 19:51:44 63.146.181.119:0 -> 10.10.152.246:0 UDP
Mar 19 19:51:43 63.146.181.119:12112 -> 10.10.152.246:30066 UDP
Mar 19 19:51:43 63.146.181.119:15677 -> 10.10.152.246:15677 UDP
Mar 19 19:51:44 63.146.181.119:7000 -> 10.10.152.246:7001 UDP
Mar 19 19:51:45 63.146.181.119:7000 -> 10.10.152.246:7001 UDP
```

Host: 211.233.25.32

```
Mar 20 22:48:21 211.233.25.32:516 -> 10.10.153.153:1588 UDP
Mar 20 22:48:25 211.233.25.32:0 -> 10.10.153.153:0 UDP
Mar 20 22:48:22 211.233.25.32:20125 -> 10.10.153.153:46286 UDP
Mar 20 22:48:23 211.233.25.32:7000 -> 10.10.153.153:7001 UDP
Mar 20 22:48:24 211.233.25.32:25793 -> 10.10.153.153:54770 UDP
```

Port 7000 and 7001 are associated with port IRC and afs3. Due to the irregularity of the ports I do not believe this is Afs traffic. Port 1588 is associated with Triquest-lm and port 516 is videotex. This traffic definitely does not fit within the realms of normal activity. The are several port 0 to port 0 UDP packets and random high port access within a second or two of each other. This does not necessarily fit the Adore Worm traffic pattern. It appears that many of these "Red Worm" are triggering on the random high port access in this communication.

Correlations:
A write up on the adore worm is available on the Sans web site http://www.sans.org/y2k/adore.htm.
An article titled ""Adore"" worm squirms in Linux systems" from April 4th on CNET discusses the worm http://news.com.com/2100-1001-255283.html?legacy=cnet

Defensive Recommendations:
An Adore worm detection and removal kit is available at http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm.
Even though Adore was believed to start spreading on April 1 2001 many of the vulnerabilities it uses data back into 2000. Patches were available to correct these problems. In fact, Adore itself secures the systems after infection by killing vulnerable services and disallowing anonymous ftp. By maintaining current system security patches as well as securing basic OS installs by not allowing anonymous ftp and turning of unnecessary services many of these problems can be avoided.

Event:FTP DoS ftpd globbing

Source of the attack:
This attack was chosen for analysis because the attack was always initiated from outside GIAC University's network. Several internal hosts were attacked. The current Snort signature base does not come with this signature by default so I must assume it is custom. Being a custom signature I do not know how prone it is to false alarms. Prudent security measures would be to assume it is in fact a true event. I would imagine due to the nature of this particular problem though, that there could be a large amount of false alarms

associated with this signature. The reason for this is that the signatures for this attack (which are not included in the Snort 1.8.4 signatures) are prone to false alarms. The closest signature for file Wu-Ftpd file globbing that I could find is below.

Signature: Wu-Ftpd File Globbing

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP wu-ftp file completion attempt {"; flags:A+; flow:to_server; content:"~"; content:"{"; reference:bugtraq,3581; classtype:misc-attack; sid:1378; rev:5;)
```

Attack was generated by:
Forty-eight different external IP addresses generated this attack directed towards twelve internal addresses. The primary victim was 10.10.153.191

Probability the source address was spoofed:
Very little, this attack requires the three-way handshake of TCP.

Description of the attack:
The original vulnerability is detailed at http://online.securityfocus.com/bid/3581
Red Hat originally made the attack public on November 27, 2001. The attack itself requires valid user credentials to login. Therefore the attack does not lend itself well to worm style attacks, unless anonymous ftp is enabled. The attack itself is based around the malformed globbing patterns (which interpret certain meta characters and allow for wild card searches) which can cause a heap overwrite condition. This allows the attacker to execute arbitrary code with the access of the Wu-Ftpd process, which is usually root.

Defensive Recommendations:
Vendor patches have been available since late 2001. Disabling anonymous access can significantly reduce the exposure to this attack.

Event: Out Of Spec Host 64.152.183.174 & Host 217.56.233.186

On March 20th host 64.152.183.174 scanned 316 hosts by sequentially scanning for the ftp service. On March 19th host 217.56.233.186 scanned 297 hosts in the same manner. The attacker used a source port of 21 and sent the connection attempts with the SYN-FIN flags sent. This attack is most likely a “script kiddie” using a noisy automated tool. However the reconnaissance was successful as host 10.10.150.139 responded to the reconnaissance. This attack was detailed earlier as “Detect # 3”.

Event: Scan TCP ports 1080, 3128, 8000, 8080

On March 21st ports 1080, 3128, 8000 and 8080 saw a dramatic increases in activity. The normal activity for those ports had been under ten per day but that amount increased to over well over 300 for the 21st. The ports 1080, 3128, and 8080 are associated strongly

with Ring Zero. According to Stephen Northcutt
http://www.sans.org/newlook/resources/IDFAQ/ring_zero.htm
Ring Zero occasionally will use port 8000 as well. These are well known proxy ports.
External host 61.132.208.63 generated 1080 different scan log entries.

General Recommendations:

Var-log consulting realizes that because GIAC University is a University it requires a very open security policy. Because of this open policy it is difficult to keep these attacks from occurring. The standard business style security implementations do not really fit because access is too restricted and it may be costly to implement. There are however, a few changes that can be done to reduce this type of activity significantly. To prevent internal hosts from being compromised by non directed worm style attacks it is necessary to maintain current vendor system patches. Regular security audits and holding administrators accountable for implementing system patches can ensure that critical systems are kept relatively current. GIAC does not appear to have a formal incident response plan. Actively monitoring IDS logs and responding when alerts occur can minimize the impact of these events through quick containment. Certain IDS signatures are good candidates for using automated IDS response techniques such as shunning, TCP Resets or routing hosts to Null. Integrating the process of security, where there is scheduled re-evaluation and improvement, structuring an incident response plan and holding administrators accountable for maintaining current patches on critical systems could offer a much improved security posture while still maintaining the open, flexible policy that is required by GIAC University.

References

[1] Ptacek, Thomas and Newsham, Timothy "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection" January, 1998. URL: http://secinf.net/info/ids/idspaper/idspaper.html

[2] Puppy, Rain Forest "A look at whisker's anti-IDS tactics" December 24, 1999. URL: http://www.wiretrip.net/rfp/pages/whitepapers/whiskerids.html

[3] Arboi, Michel and Deraison, Renaud "Using Nessus's NIDS evasion features" 2002, URL: http://www.nessus.org/doc/nids.html

[4] Handley, Mark Paxson, Vern and Kreibich, Christian "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics" May 22, 2001. URL: http://www.icir.org/vern/papers/norm-usenix-sec01-html

[5] Song, Dug "Fragroute", April, 2002. URL: http://www.monkey.org/~dugsong/fragroute/

[6] Miller, Toby "Passive OS Fingerprinting: Details and Techniques", November 6, 2001, URL: http://www.incidents.org/papers/OSfingerprinting.php.

Appendix A: Splicer.pl

```
#!/usr/bin/perl

use Socket;
use Getopt::Std;

require 'getopts.pl';
getopts('h:t:s:r:');
if (!$opt_h || !$opt_t || !$opt_r) {
	&usage();
	exit 1;
}

sub usage {
	print "\tUSAGE: splicer -h <host> -r <request> -t <timing> -s<splice sizes>\n\n";
	print "\t***********Notes**********\n";
	print "\t***Apache times out a unfinished request in 6 minutes\n";
	print "\t***IIS doesn't appear to have such a timeout\n";
	print "\t***If no splice size is specified it will send 1 byte at a time\n\n";
}

if ($opt_s == NULL) {
	$opt_s=1;
}

$host=$opt_h;
$size=$opt_s;
$req="GET $opt_r HTTP/1.0\r\n";
$time=$opt_t;

$header="Host: $host\r\nUser-Agent: Session-Splice\r\n\r\n";


@greq=(split//,$req);
socket(SERVER, PF_INET, SOCK_STREAM, getprotobyname('tcp'));
$addr = sockaddr_in(80, inet_aton($host));
connect(SERVER, $addr);
select(SERVER); $|=1;
select(STDOUT);
$i=0;

# Send packet
foreach $char (@greq) {
	chomp $char;
```

```
    if ($i == $size) {
    push(@new,$char);
    print SERVER @new;
    select(undef,undef,undef,$time);
    $i = 0;
    $i++;
    $#new=-1;
    } else {
        push(@new,$char);
        $i++;


    }
}
print SERVER "$header";

# Parse our requesr
do {
    $line = <SERVER>
    }
    until ($line =~ /^\r\n/);
    @output = <SERVER>;
    close (SERVER) ;
    print "@output\n";
```

Appendix B: Cisco IDS Log Format

Cisco Log Description

Field	Description	Value
1	Record Type	4
2	Record ID	1403640
3	GMT Datestamp	2002/03/08
4	GMT Timestamp	17:51:01
5	Local Datestamp	2002/03/08
6	Local Timestamp	9:51:01
7	Application ID	10008
8	Host ID	X
9	Organization ID	X
10	Source Direction	OUT
11	Destination Direction	IN
12	Alarm Level	4
13	Sig ID	8000
14	SubSig ID	8010
15	Protocol	TCP/IP
16	Source IP	198.83.130.39
17	Destination IP	X.X.X.X
18	Source Port	2223
19	Destination Port	80
20	Router IP	0.0.0.0
21	Data	

Appendix C: **tcp_report.pl and udp_report.pl**

tcp_report.pl

```
#!/usr/bin/perl

open (outfile, ">tcp_report.csv");
$t18=`cat 18.tcp.int|wc -l`;
$t19=`cat 19.tcp.int|wc -l`;
$t20=`cat 20.tcp.int|wc -l`;
$t21=`cat 21.tcp.int|wc -l`;
$t22=`cat 22.tcp.int|wc -l`;
$t23=`cat 23.tcp.int|wc -l`;

@portlist=`cat tcpports.int`;
@tcp18=`cat 18.tcp.int.srt`;
@tcp19=`cat 19.tcp.int.srt`;
@tcp20=`cat 20.tcp.int.srt`;
@tcp21=`cat 21.tcp.int.srt`;
@tcp22=`cat 22.tcp.int.srt`;
@tcp23=`cat 23.tcp.int.srt`;

foreach $line(@tcp18) {
chomp $line;
for ($line) {
s/^\s+//;
}
($count,$port)=split(/\s+/,$line);
$h18{$port} = $count;
}

foreach $line(@tcp19) {
chomp $line;
for ($line) {
s/^\s+//;
}
($count,$port)=split(/\s+/,$line);
$h19{$port} = $count;
}

foreach $line(@tcp20) {
chomp $line;
for ($line) {
s/^\s+//;
}
($count,$port)=split(/\s+/,$line);
```

```
$h20{$port} = $count;
}

foreach $line(@tcp21) {
chomp $line;
for ($line) {
s/^\s+//;
}
($count,$port)=split(/\s+/,$line);
$h21{$port} = $count;
}

foreach $line(@tcp22) {
chomp $line;
for ($line) {
s/^\s+//;
}
($count,$port)=split(/\s+/,$line);
$h22{$port} = $count;
}

foreach $line(@tcp23) {
chomp $line;
for ($line) {
s/^\s+//;
}
($count,$port)=split(/\s+/,$line);
$h23{$port} = $count;
}

foreach $p(@portlist) {
chomp $p;
#print "$p\n";
#print "$h18{$p}\n";
$d18=$h18{$p};
$d19=$h19{$p};
$d20=$h20{$p};
$d21=$h21{$p};
$d22=$h22{$p};
$d23=$h23{$p};

$p18=(sprintf "%.2f",($d18/$t18));
$p19=(sprintf "%.2f",($d19/$t19));
$p20=(sprintf "%.2f",($d20/$t20));
$p21=(sprintf "%.2f",($d21/$t21));
```

```
$p22=(sprintf "%.2f",($d22/$t22));
$p23=(sprintf "%.2f",($d23/$t23));

$tot=($p18+$p19+$p20+$p21+$p22+$p23);
$a=($tot/6);
$max=($a*3);
if (($p18 > $max)||($p19 > $max)||($p20 > $max)||($p21 > $max)||($p22 > $max)||($p23
> $max))  {
        print outfile
"$p,$d18,$p18,$d19,$p19,$d20,$p20,$d21,$p21,$d22,$p22,$d23,$p23\n";
        } else {}
}
close(outfile);
```

udp_report.pl

```
#!/usr/bin/perl

open (outfile, ">udp_report.csv");
$t18=`cat 18.udp.int|wc -l`;
$t19=`cat 19.udp.int|wc -l`;
$t20=`cat 20.udp.int|wc -l`;
$t21=`cat 21.udp.int|wc -l`;
$t22=`cat 22.udp.int|wc -l`;
$t23=`cat 23.udp.int|wc -l`;

@portlist=`cat udpports.int`;
@udp18=`cat 18.udp.int.srt`;
@udp19=`cat 19.udp.int.srt`;
@udp20=`cat 20.udp.int.srt`;
@udp21=`cat 21.udp.int.srt`;
@udp22=`cat 22.udp.int.srt`;
@udp23=`cat 23.udp.int.srt`;

foreach $line(@udp18) {
chomp $line;
for ($line) {
s/^\s+//;
}
($count,$port)=split(/\s+/,$line);
$h18{$port} = $count;
}

foreach $line(@udp19) {
chomp $line;
```

```
for ($line) {
s/^\s+//;
}
($count,$port)=split(/\s+/,$line);
$h19{$port} = $count;
}

foreach $line(@udp20) {
chomp $line;
for ($line) {
s/^\s+//;
}
($count,$port)=split(/\s+/,$line);
$h20{$port} = $count;
}

foreach $line(@udp21) {
chomp $line;
for ($line) {
s/^\s+//;
}
($count,$port)=split(/\s+/,$line);
$h21{$port} = $count;
}

foreach $line(@udp22) {
chomp $line;
for ($line) {
s/^\s+//;
}
($count,$port)=split(/\s+/,$line);
$h22{$port} = $count;
}

foreach $line(@udp23) {
chomp $line;
for ($line) {
s/^\s+//;
}
($count,$port)=split(/\s+/,$line);
$h23{$port} = $count;
}

foreach $p(@portlist) {
chomp $p;
```

```
#print "$p\n";
#print "$h18{$p}\n";
$d18=$h18{$p};
$d19=$h19{$p};
$d20=$h20{$p};
$d21=$h21{$p};
$d22=$h22{$p};
$d23=$h23{$p};

$p18=(sprintf "%.6f",($d18/$t18));
$p19=(sprintf "%.6f",($d19/$t19));
$p20=(sprintf "%.6f",($d20/$t20));
$p21=(sprintf "%.6f",($d21/$t21));
$p22=(sprintf "%.6f",($d22/$t22));
$p23=(sprintf "%.6f",($d23/$t23));

$min=".00009";
if (($p18 < $min)||($p19 < $min)||($p20 < $min)||($p21 < $min)||($p22 < $min)||($p23 <
$min)) {
        } else {
$tot=($p18+$p19+$p20+$p21+$p22+$p23);
$a=($tot/6);
$max=($a*2);
if (($p18 > $max)||($p19 > $max)||($p20 > $max)||($p21 > $max)||($p22 > $max)||($p23
> $max))  {
        print outfile
"$p,$d18,$p18,$d19,$p19,$d20,$p20,$d21,$p21,$d22,$p22,$d23,$p23\n";
        } else {}
        }
}
close(outfile);
```

Appendix D: Methodologies

The methodologies were quit simple. I first attempted SnortSnarf, which due to the large amount of data promptly caused my personal server to crash on three occasions. Realizing that wasn't getting me anywhere I resorted to standard Unix tools like sed, awk, cut, grep and vi. For ease in sorting or trying to make sense out of MY.NET I converted every IP that was MY.NET to a 10.10 network. This was done using the command "sed s/MY.NET/10.10/g". I created a simple shell script called sort.sh to help sort the data.

Sort.sh

```
#!/bin/sh

cat $1 |awk -F\> '{print $2}' |awk -F: '{print $2}' |cut -d" " -f 1 |sort |uniq -c |sort -nr>
$1.srt
```

I used a variety of variants to distill the data. I then wrote the two perl scripts tcp_report.pl and udp_report.pl to establish port relationships over time. I tweaked the variables in those scripts until my output was eight to twelve events, which was enough to make a clean graph. I generated another graph to depict threat source through use of the common Unix tools alluded earlier.