Global Information Assurance Certification Paper

GIAC Training and Certification

Track 3 – Intrusion Detection In-Depth
GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment
Version 4.0

Jan Stodola

September 23, 2004

Table of Content

1 Executive Summary

Atrix Network Consulting (ANC) is a privately held network security company, mandated with security audit of ABC University network logs. The core data, collected by industry standard Snort tool, dates around October 18, 2002. The supplementary data span October 14-17, 2002.

The University network seems to be in a very good condition. ANC found no sign of compromise or anomalous behavior. But there are several areas of concern that would improve the overall network security posture.

Understandably the University is expected to promote information access and sharing. Unfortunately the Internet is becoming an increasingly hostile environment, threatening integrity and availability of every network connected to it. ANC discovered that large volume of malware traffic is permitted from Internet to the inside perimeter of the University network. This finding may be an opportunity to review the IT networking Mission Statement to provide framework for a balanced level of incoming traffic filtering.

ANC identified significant outbound Instant Messaging (IM) traffic in form of MSN Messenger chat. IM is a very vulnerable application group from the security standpoint. As such, the Acceptable Network Usage Policy needs to be very specific on permitted IM clients, their use and conditions a computer host has to meet to be allowed to run IM client. Due to permissive nature of University network environment, the IT department may wish also to focus on approach that allows quick recovery from host infection.

Lastly ANC would like to highlight need for a comprehensive network security infrastructure upgrade. To facilitate thorough forensic analysis, traffic and key infrastructure points (such as firewalls) log data need to be collected on the perimeter and inside of the University network. Along with the existing one, new network and host based Intrusion Detection Systems are necessary. These, combined with around-the-clock Computer Incident Response Team organizational and technological infrastructure, are needed for quality continuous network security health monitoring and timely handling of network security incidents.

More specific network security recommendations can be found in the section 2.9 Defensive Recommendations.

2 Detailed Analysis

The analytical approach and associated tools are presented in the section Analysis Process.

2.1 Scenario Identification

Logs from ABC University had been downloaded from http://isc.sans.org/logs/Raw/. The core data analyzed are in file 2002.10.18. Included datagrams do not match the file name - span between 00:03:26 on November 17, 2002 and 09:45:22 on November 18, 2002, unknown time zone, assumed to be EST. As per associated README file, the

data is binary output from a Snort instance, with sanitized internal IP address space and modified check sums. Additionally, all ICMP, DNS, SMTP and Web traffic has been removed. The collection rule set is unknown. For detect correlation we extended the scope with files 2002.10.14 to 2002.10.17, containing the same internal network subnet IP address range of 170.129.0.0/16.

2.2 Relationship Analysis

2.2.1 Basic Traffic Anatomy

Unique MAC source and destination addresses in the raw log file:

```
00:00:0c:04:b2:33
00:03:e3:d9:26:c0
```

We are examining data from a tap/span port monitoring traffic between two L2 (Ethernet) devices. We found no datagram with the same source and destination MAC address – no anomaly in this respect. The device manufacturer can be found on IEEE[1] site as Cisco in both cases.

Next we determine source and destination IP addresses and destination ports associated with traffic originated from both L2 points. (The values in a specific row are not associated – we present three independent columns of values):

Source IP	Destination IP	Destination Port
128.167.120.13 153.33.24.3 161.69.201.238 … 170.129.15.162 170.129.21.101 170.129.21.111 170.129.21.117 170.129.21.122 170.129.21.128 170.129.21.133 170.129.21.138 170.129.21.144 170.129.21.149 170.129.21.154 170.129.21.160 … 80.7.188.43 80.7.32.154 81.98.99.83	170.129.100.243 170.129.108.132 170.129.113.233 … 170.129.48.119 170.129.50.11 170.129.50.120 170.129.50.16 170.129.50.3 170.129.57.187 … 170.129.86.252 170.129.91.105 170.129.91.211	0 80 111 139 515 1080 1839 3128 4343 8080 61004 61146 61147 … … 64946 65018 65025
Total: 96	Total: 80	Total: 75

Table 1 Traffic Associated with Source MAC Address 00:03:e3:d9:26:c0

Range of 96 unique source IP addresses originating from 00:03:e3:d9:26:c0 (we call this device Cisco-1) is a diverse set of public addresses, but also contains some destination network address space (examined below). There are 80 unique destination

[1] IEEE.

IP addresses; all of them with prefix 170.129.0.0/16. Note large number of different services (75) in this traffic. In total Cisco-1 forwards 319 datagrams.

Source IP	Destination IP	Destination Port
170.129.50.3 170.129.50.120	144.9.72.134 164.109.22.53 194.67.23.251 194.67.35.196 … … 66.135.192.83 66.35.229.102 66.35.229.104 66.77.49.240	80 1709 1863 2150
Total: 2	**Total: 64**	**Total: 4**

Table 2 Traffic Associated with Source MAC Address 00:00:0c:04:b2:33

On the other hand traffic originated from the device with MAC address 00:00:0c:04:b2:33 (we call it Cisco-2) shows only two unique source IP addresses in 170.129.50.0/16 range, four destination port but 74 destination IP addresses. In total Cisco-2 originates (as forwarder) 1365 datagrams.

Cisco-1 seems to face Internet or intranet close to the Internet as it connects many diverse source IP addresses to a few destination IP addresses. The traffic from Cisco-2 presents leaner source IP range - 170.129.50.0/24 – so we believe it is the access point to the internal network. Asymmetry in the destination IP address and TCP port ranges of both devices complement this observation. Figure 2 shows the network topology. Based on above traffic asymmetry, we also believe that the monitoring point is in a DMZ-like environment.

So many ports open on Cisco-1 suggest it likely not to be a firewall. Lets assume it is a (border) router. Rob Perdue[2] found a PIX firewall with MAC address lower that CISCO-1. We were unable to locate PIX with higher MAC address, to finish the MAC triangulation and support this assertion.

Twelve ingress datagrams with source IP address in the range of destination B class (Table 1) looked at first as a spoofing attempt. Then Ethereal showed them as complete IGMP (Internet Group Management Protocol) V2 Membership Query. Eric Hall[3]: "Membership Query messages are sent by multicast routers whenever they want to verify that hosts are listening for remotely generated multicast traffic that is being forwarded to this network." These queries can be interpreted as indication that behind Cisco-2 are hosts in subnets 170.129.15.0/24 and 170.129.21.0/24. (Brett Hutley[4] makes a case for these IGMP datagrams to be crafted – always a possibility impacting the network layout assessment.)

[2] Perdue, Rob.
[3] Hall, Eric - p.159.
[4] Hutley, Brett.

Cisco-1 permits 190 datagrams of ingress traffic to 170.29.0.0/16 other than 170.29.50.0/24. The IGMP traffic above may very well be sanctioned. But most of other traffic has a dubious purpose/content. For example traffic to unusual "well known" ports (IANA[5]: 0-reserved, 111-Sun RPC, 139-NETBIOS and 515-printer spooler – examined in 2.4 Detect 2 – BACKDOOR Q) is of interest. The log venue point does not allow seeing any inside traffic of the internal network.

The log file Snort detect totals are listed in Appendix 4.2.

2.2.2 Web Server

The internal IP address 170.129.50.3 manifests its active presence with only two datagrams and associated "ATTACK-RESPONSES 403 Forbidden" detects. Here is pertinent part of the first datagram, as presented by tcpdump:

```
04:28:13.946507 00:00:0c:04:b2:33 > 00:03:e3:d9:26:c0, ethertype IPv4
(0x0800), length 552: IP (tos 0x0, ttl  63, id 63, offset 0, flags [DF], length: 538)
170.129.50.3.80 > 195.29.139.29.2150: P [bad tcp cksum 1634 (->3974)!] 3259418630
:3259419116(486) ack 3087187806 win 31856 <nop,nop,timestamp 2983552 2014628>
        0x0000:  4500 021a 003f 4000 3f06 0ee0 aa81 3203  E....?@.?.....2.
        0x0010:  c31d 8b1d 0050 0866 c246 c806 b802 bf5e  .....P.f.F.....^
        0x0020:  8018 7c70 1634 0000 0101 080a 002d 8680  ..|p.4.......-..
        0x0030:  001e bda4 4854 5450 2f31 2e31 2034 3033  ....HTTP/1.1.403
        0x0040:  2046 6f72 6269 6464 656e 0d0a 4461 7465  .Forbidden..Date
        0x0050:  3a20 4d6f 6e2c 2031 3820 4e6f 7620 3230  :.Mon,.18.Nov.20
        0x0060:  3032 2031 333a 3138 3a32 3220 474d 540d  02.13:18:22.GMT.
        0x0070:  0a53 6572 7665 723a 2041 7061 6368 652f  .Server:.Apache/
        0x0080:  312e 332e 3132 2028 556e 6978 2920 2028  1.3.12.(Unix)..(
        0x0090:  5265 6420 4861 742f 4c69 6e75 7829 2046  Red.Hat/Linux).F
        0x00a0:  726f 6e74 5061 6765 2f34 2e30 2e34 2e33  rontPage/4.0.4.3
        0x00b0:  0d0a 436f 6e6e 6563 7469 6f6e 3a20 636c  ..Connection:.cl
        0x00c0:  6f73 650d 0a43 6f6e 7465 6e74 2d54 7970  ose..Content-Typ
(log truncated)
```

Source port number 80 and payload content are associated with a Web server. In the payload we can see the identification "Server: Apache/1.3.12 (Unix) (Red.Hat/Linux) FrontPage/4.0.4.3". Full description of the authoring application is FrontPage Server Extension (FPSE). As per Microsoft[6] the version running on Apache 1.3.12 can be only FPSE 2000. FPSE 2002 would require Apache 1.3.19[7]. TTL value 63 is one less than default 64 expected from UNIX host, as per p0f[8] fingerprint database file p0f.fp. One hop delta from potential default indicates that the Web server may be connected directly to Cisco-2.

As per available log file, the server did not respond to any stimuli, (seen in link graph on Figure 1). This is an indication that the server is not susceptible to the incoming exploits (IIS, FrontPage and formmail). The only outgoing traffic were two notifications of access denial to a specific web page – a normal response from a Web server. The responses are directed to the requesting ports 1709 and 2150, accounting for occurrence of these

[5] IANA.
[6] Microsoft [1].
[7] Microsoft [2].
[8] p0f.

two ports in Table 2. Hence the Web server does not actively target these ports. There are no logs of requests triggering these responses. The Web server does not generate any other detects – a sign it is infection free. Worth noting is that the Web server is patched for exploit associated with ingress FPSE detect (see section 2.3.5).

For the link graph we selected traffic on the Web server – the most important network asset found. From many interesting exhibits in the graph we mention just incoming HTTP acknowledgements from Web server port 80 to Web server port 80 – unusual source port for client-server protocol.

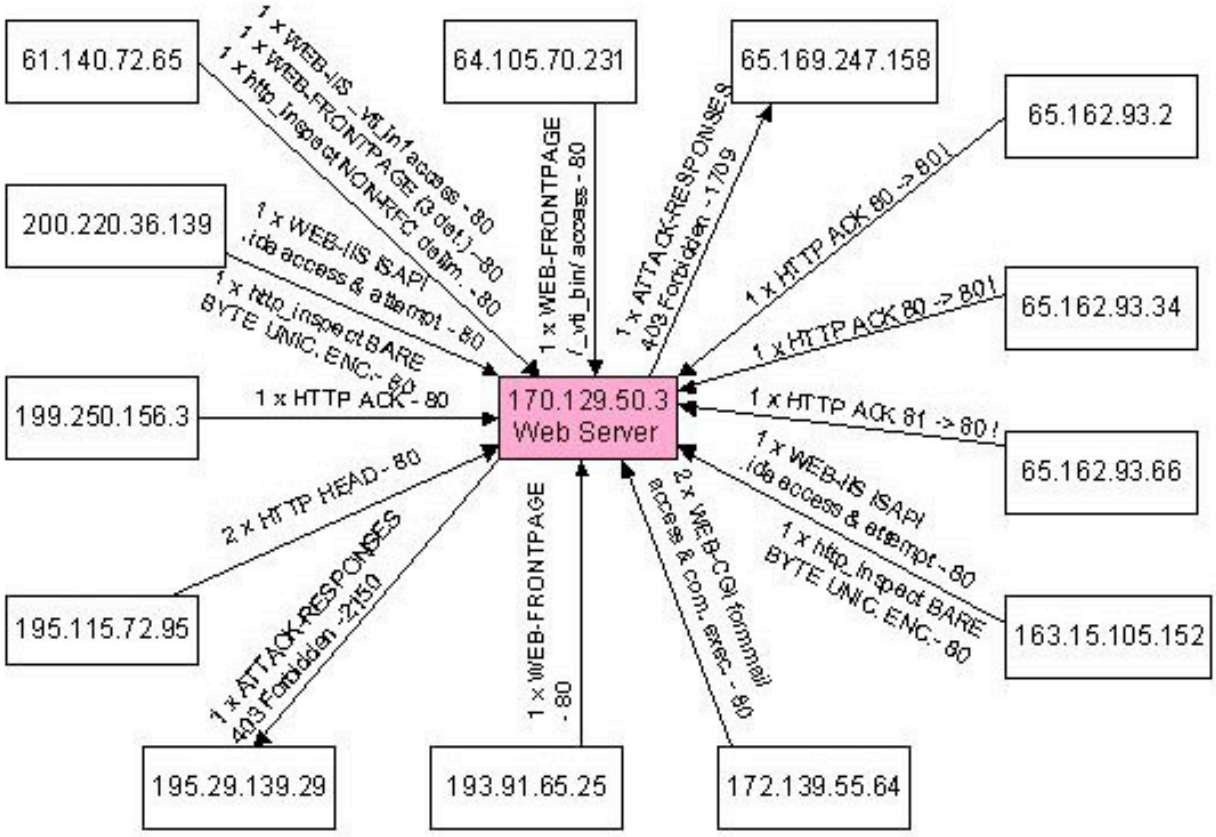

Figure 1 170.129.50.3 Web Server Link Graph

2.2.3 Cisco-2 and Internal Network

The majority of traffic, coming from protected network through Cisco-2 IP address 170.129.50.120, is directed to various external Web servers. Lets examine payload of egress GET requests:

User-Agent Field Content	TTL
Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0 YComp 5.0.2.6)	124
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)	125
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 4.0)	124
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)	123
Mozilla/4.0 (compatible; MSIE 4.01; Windows NT)	124

Table 3 Cisco-2 User-Agent and TTL Values of Outgoing HTTP GET Requests

Variations in User-Agent HTTP field show that there are different hosts making requests through one external IP address. IP TTL value variations not only provide confirmation of this hypothesis, but also indicate in that the hosts most may vary in hop distance from Cisco-2. In other words, the internal network likely contains several routers.

NAT (Network Address Translation) is one of ways to provide this “invisibility” of internal network IP address structure. Taking in consideration no egress response to ingress detects (sign of packet filtering taking place), Cisco-2 can be one of several products, like a firewall or a router with a tight ACL. We did not succeed with a close MAC address triangulation to associate the device with a specific hardware platform.

There are 122 detects against Cisco-2 with targets other than the Web server above – scans, information requests and malformed traffic of various shape. The internal network did not respond to any of these stimuli. It also did not generate any malware related egress detects – sign of good health of internal hosts.

The remaining Cisco-2 egress traffic listed in Table 2 is limited to originator 170.129.50.120, directed at port 80 (normal Web traffic, no anomalies noted) and 1863 (MSN Instant Messaging traffic). There are 169 “CHAT MSN message” detects associated with destination port 1863, identified as user chat, as analyzed in section 2.5.

2.2.4 Putting it All Together

Above discussion can be summarized in the following network topology:

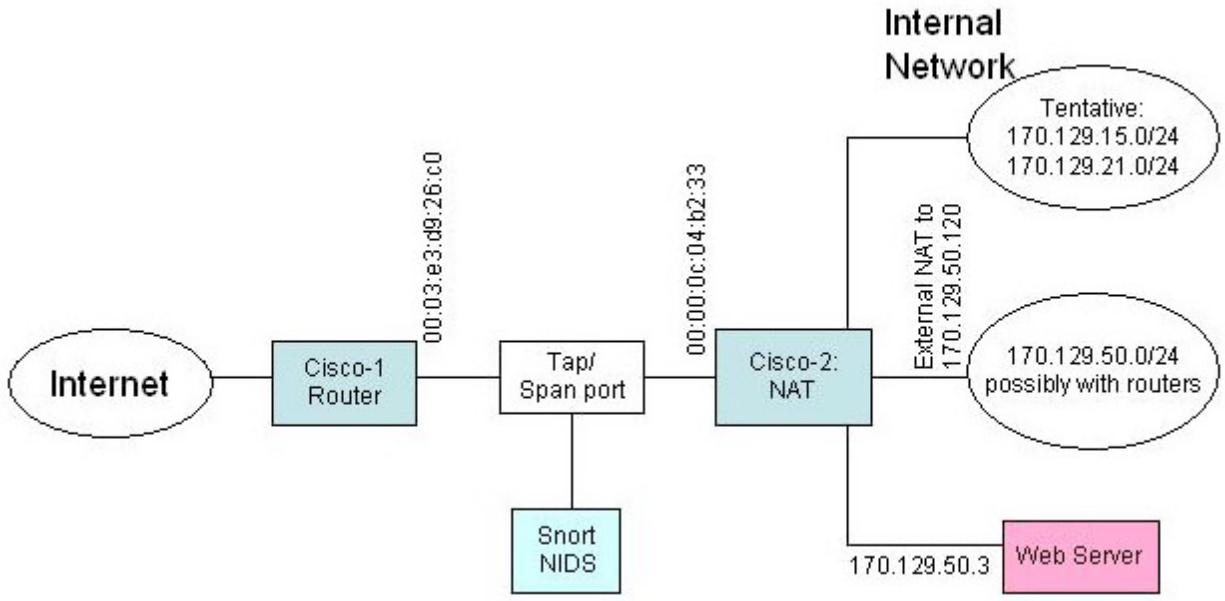

Figure 2 Network Topology

Missing Detects
There were 308 detects generated from 1685 datagrams in the log file. That is relatively low count – the log file is after all output of Snort instance.

The biggest disproportion is in egress traffic. All datagrams to egress ports 1709, 1863 and 2150 in Table 2 are associated with detects – 171 in total. But there are no detects associated with any of 1195 datagrams towards external Web server destinations on port 80. Some of the "missing" detects may be attributed to HTTP tuning of our Snort instance, as described in 3.2 Snort Tuning. Such tuning may not have taken place in the collection rule set. Other explanation may be use of customized rules in collecting Snort instance. Triggering rules and datagram sequencing do not suggest use of tagging to explain datagram excess.

Looking at this egress Web traffic, we see a couple of themes, in different datagrams:
- Cookies: large size (>1500 bytes), non-ASCII content
- De-compression: "Accept-encoding:" field values gzip, deflate in HTTP continuation datagrams
- De-compression: "Accept-encoding:" field values gzip, deflate in HTTP GET command to a suspicious site cooking.ru.

Further investigation would be needed to progress on this subject.

2.3 Detect 1 - WEB-FRONTPAGE _vti_rpc access

```
[**] [1:937:7] WEB-FRONTPAGE _vti_rpc access [**]
[Classification: access to a potentially vulnerable web application]
[Priority: 2] 11/17/02-20:33:20.336507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
len:0x201 61.140.72.65:51597 -> 170.129.50.3:80 TCP TTL:45 TOS:0x0 ID:20630 IpLen:20
DgmLen:499 DF ***AP*** Seq: 0xB47BE106  Ack: 0xC12ECE7B  Win: 0x3890  TcpLen: 32 TCP
Options (3) => NOP NOP TS: 84578379 134194 [Xref =>
http://www.securityfocus.com/bid/2144]
```

2.3.1 Description of Detect

From Snort sid 937[9]: "This event is generated when an attempt is made to exploit a known vulnerability in a Web server running Microsoft FrontPage Server Extensions." Bugtraq 2144[10] elaborates:

> Due to the way FPSE handles the processing of Web forms, IIS is subject to a denial of service. By supplying malformed data to one of the FPSE functions IIS will stop responding. A restart of the service is required in order to gain normal functionality. It should be noted that the victim only requires to have FPSE installed on the Web server to be vulnerable.

Impact as per Snort sid 937: "Information gathering and system integrity compromise. Possible unauthorized administrative access to the server or application. Possible

[9] Snort [3].
[10] SecurityFocus [1].

execution of arbitrary code…" Sid 937also confirms that all systems running FPSE are vulnerable.

The only possible specific impact in our UNIX server scenario was found in the FPSE 2000 Service Release 1.2 list[11] of addressed issues, which states in the UNIX section: "An external attack could cause the server's CPU to spike".

This vulnerability is associated with CVE-2001-0096.[12]

2.3.2 Reason This Detect Was Selected

- Directed against the most critical internal asset we enumerated: the Web server.
- Evidence of active targeting.

2.3.3 Detect Was Generated By

The raw data context in 2.1 Scenario Identification was processed as described in 3.4 Detect Generation and Analysis. Section 4.3 FrontPage Session Detects and Datagrams contains all available detects and datagrams of the associated FrontPage session. This detect was triggered by following rule from web-frontpage.rules file:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-FRONTPAGE
_vti_rpc access"; flow:to_server,established; uricontent:"/_vti_rpc"; nocase;
reference:bugtraq,2144; classtype:web-application-activity; sid:937; rev:7;)
```

The rule verbatim: Raise an alert on TCP traffic originating from any port of external network (set to any IP address in snort.conf). We can see the destination as a HTTP port (set to 80 in snort.conf) on HTTP server (specified to 170.129.50.3 in snort conf). The TCP connection has to be already established and the rule shall be triggered only on traffic towards Web server. The case non-sensitive string "/_vti_rpc" shall be part of normalized URI (Universal Resource Identifier) buffer sent by the Web client. Bugtraq item 2144 is provided serves as reference. The rule class is determined as web-application-activity. Number 937 is the rule identification, 7th revision.

How the triggering datagram correlates with the rule:

```
20:33:20.336507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4
(0x0800), length 513: IP (tos 0x0, ttl  45, id 20630, offset 0, flags [DF], length:
499) 61.140.72.65.51597 > 170.129.50.3.80: P [bad tcp cksum 99c5 (->bd05)!]
3028017414:3028017861(447) ack 3241070203 win 14480 <nop,nop,timestamp 84578379
134194>
        0x0000:  4500 01f3 5096 4000 2d06 991d 3d8c 4841  E...P.@.-...=.HA
        0x0010:  aa81 3203 c98d 0050 b47b e106 c12e ce7b  ..2....P.{.....{
        0x0020:  8018 3890 99c5 0000 0101 080a 050a 904b  ..8............K
        0x0030:  0002 0c32 504f 5354 202f 5f76 7469 5f62  ...2POST./_vti_b
        0x0040:  696e 2f73 6874 6d6c 2e65 7865 2f5f 7674  in/shtml.exe/_vt
        0x0050:  695f 7270 6320 4854 5450 2f31 2e30 0d0a  i_rpc.HTTP/1.0..
        0x0060:  4461 7465 3a20 5765 642c 2031 3620 4a61  Date:.Wed,.16.Ja
        0x0070:  6e20 3230 3032 2030 353a 3337 3a31 3720  n.2002.05:37:17.
        0x0080:  474d 540d 0a4d 696d 652d 5665 7273 696f  GMT..Mime-Versio
```

[11] Microsoft [2] – follow link in sentence: "See the list of issues addressed with Service release 1.2…"
[12] CVE [2].

```
        0x0090:  6e3a 2031 2e30 0d0a 5573 6572 2d41 6765  n:.1.0..User-Age
        0x00a0:  6e74 3a20 4d53 4672 6f6e 7450 6167 652f  nt:.MSFrontPage/
        0x00b0:  342e 300d 0a41 6363 6570 743a 2061 7574  4.0..Accept:.aut
        0x00c0:  682f 7369 6369 6c79 0d0a 436f 6e74 656e  h/sicily..Conten
        0x00d0:  742d 4c65 6e67 7468 3a20 3431 0d0a 436f  t-Length:.41..Co
        0x00e0:  6e74 656e 742d 5479 7065 3a20 6170 706c  ntent-Type:.appl
        0x00f0:  6963 6174 696f 6e2f 782d 7777 772d 666f  ication/x-www-fo
        0x0100:  726d 2d75 726c 656e 636f 6465 640d 0a58  rm-urlencoded..X
(datagram truncated)
```

The byte on offset 9th (0x06) matches TCP protocol. Source address and port fits by default to "any" value. Destination address on offset 16-19 (0xaa813203 or 170.129.50.3) is covered by 170.129.0.0/16. Destination port on 22-23th byte (0x50 or 80) fits the rule. TCP ACK and PSH flag are on in byte 33 (0x18), indicating established TCP connection (absence of SYN, FIN or RST flags). The Web server position as the destination address satisfies the condition of datagram directionality towards server. The Web client request string "/_vti_rpc" starting at offset 76 meets the URI content requirement.

2.3.4 Probability the Source Address Was Spoofed

The source address (located on extranet, as per its source MAC address) was not spoofed. The log file contains in total 3 datagrams of an established TCP session originating at 61.140.72.65, spanning 2.5 seconds. (The second datagram triggers examined WEB-FRONTPAGE vti_rpc access event). It would be close to impossible to keep datagram timing and acknowledgements correct in spoofed scenario for such a long period of time without breaking the connection.

Additional points:

- TTL value of all datagrams is consistent and reasonable at 45.
- Given 2.5 seconds delay, ID difference of 4 between first and second datagram (see section 2.3.5) is reasonable within one TCP session.
- The GET /_vti.inf.html command seen as the very first FrontPage session packet is expected to return valuable information. The attacker can easier see that if the source address in not spoofed.
- FPSE is designed for remote access. It does not have concept of trust or not trust to certain machines, assuming they are served in the same virtual directory. This removes one of motivations for address spoofing.
- FPSE session timing and content is very similar to one examined by John Jetmore (see section 2.3.5), confirming it's not spoofed, non-malicious nature.

2.3.5 Attack Mechanism

The exploit works by sending very large POST request to the Web server, creating a buffer overflow situation.

John Lampe developed Nessus plugin script[13] for IIS to send about 5kB of data in POST /_vti_bin/shtml.dll/_vti_rpc. He specified "Content-Length: 5058". Server

[13] Lampe, John.

acceptance of such a large post with response id 200 (Web Ok) would confirm vulnerability.

Lets put it in the context of the investigated detect: First “GET./_vti_inf.html” request with ID 20626 is recorded, triggering “WEB-IIS vti_inf access” detect. Two and half seconds later a datagram with HTTP payload “POST /_vti_bin/shtml.exe/_vti_rpc HTTP/1.0” arrives (IP ID 20630), triggering “WEB-FRONTPAGE shtml.exe access”, “WEB-FRONTPAGE /vti_bin/ access” and “WEB-FRONTPAGE vti_rpc access” detects. The sequence number of the second datagram 20630 is by four higher then the ID of the first one, indicating that there may be other interleaving datagrams missing in our log (or datagrams unrelated to this session consumed the missing IDs).

At the same time arrives a short continuation datagram (41 bytes of HTTP payload “method=server+version:4.0.2.2611.%2e2611 ”, IP ID incremented by one to 20631), triggering “(http_inspect) NON-RFC HTTP DELIMITER” detect. No other log data for this source IP address is available, in either traffic direction.

The “Length: 41” attribute (bold in 2.3.2) in detect datagram represents the length of data submitted. It matches the HTTP payload on the subsequent packet. The size of 41 bytes is definitely consistent and reasonable, not a buffer overflow attempt. We can conclude that this detect is a false positive. (The rule does not interrogate length of the POST request. Including the length of POST in the rule, or other rule tightening, is recommended to reduce probability of false positives.)

The POST in the IIS script by John Lampe contains “/shtml.dll/” path, vice “/shtml.exe/” in the examined datagram. So does the later format constitute valid path and action? The answer is yes: John Jetmore recorded FrontPage posting session[14] on UNIX 1.3.14 / FPSE 2000 server, containing “/shtml.exe/” POST path. John later concluded[15] that the session itself was a regular FrontPage transaction.

In 2.2.2 Web Server section we established that the FrontPage 2000 version is 4.0.4.3. Microsoft fixed this buffer overflow vulnerability in FPSE 2000 for both Windows and UNIX version 4.0.2.4222[16]. Hence the examined Web server is already patched against this overflow.

It is possible to generate an isolated datagram with this overflow – for example by a recompiled version of gspoof[17] tool. The recompilation is needed to extend TCP payload beyond 128 bytes. But this solution would not work for generating the whole FPSE session.

Only one WEB-FRONTPAGE vti_rpc access detect was recorded on in the 2002.10.18 log.

[14] Jetmore, John [1].
[15] Jetmore, John [2].
[16] Microsoft [1].
[17] Embyte.

2.3.6 Correlations

See references in the text above. The vulnerability was described in BugTraq ID 2144[18]. Microsoft released a Security Buletin MS-100 on the subject http://www.microsoft.com/technet/security/bulletin/MS00-100.mspx and a patch Service Release 1.2 http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnservext/html/sr12.asp for ISS and UNIX servers.

We did not find any GCIA Practical detect on this subject. The closest one was "WEB-IIS _vti_inf access" (triggered by the first datagram of our session) by Julien Radoff: http://www.giac.org/practical/GCIA/Julien_Radoff_GCIA.zip.

This vulnerability is associated with CVE-2001-0096.[19]

2.3.7 Evidence of Active Targeting

Evidence at hand indicates intent of active targeting against the Web server. The 2002.10.18 log contains only one session (three datagrams) to originate from 61.140.72.65. No other asset than the Web server was targeted. The "GET /_vti_inf.html" datagram can be qualified as a (confirmation) reconnaissance. There is no datagram originated from 61.140.72.65 in the review period of files 2002.10.14 to 2002.10.17, adding weight to the notion of active targeting.

2.3.8 Severity = 2

Severity – (criticality + lethality) – (system countermeasures + network countermeasures)
Scale for each from 1 (lowest) to 5 (highest).

(5 + 5) – (5 + 3) = 2

Criticality	The target is a Web server. In the today's world, Corporate image and revenue may be seriously affected by unavailability, defacement or information leak of the Web server.	5
Lethality	Microsoft lists DoS as impact, Snort mentions possibility of administrative access and arbitrary code execution (not confirmed on UNIX server). Possible active targeting worsens the situation.	5
System Countermeasures	We have a solid evidence of fully patched Web server.	5
Network Countermeasures	There is no proof of a network side Web server protection layer. But we found (other subnet) egress traffic filtering as evidence of present compounded layers of defense.	3

Table 4 Severity of WEB-FRONTPAGE _vti_rpc access

[18] SecurityFocus [1].
[19] CVE [2].

2.4 Detect 2 – BACKDOOR Q access

```
[**] [1:184:6] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
11/17/02-20:11:56.686507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
255.255.255.255:31337 -> 170.129.166.76:515 TCP TTL:14 TOS:0x0 ID:0 IpLen:20
DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]
```

2.4.1 Description of Detect

As per Honeypot Project[20]:

> Q v2.0 is a client / server backdoor, featuring remote shell access with strong encryption for root and normal users and a encrypted on-demand tcp relay/bouncer that supports encrypted sessions with normal clients using the included tunneling daemon. It also has stealth features like activation via raw packets, syslog spoofing and single on-demand sessions with variable ports.

The Q contains three basic components: one way stealth messenger (qs), who carries activation vector to server (qd) running on the infected host. The server is then ready to communicate with the commanding client (q).

Although primarily a UNIX trojan, Les Gordon[21] points out that it may be possible to compile and run the Q in Windows environment. Les also writes: “All versions of Q are compiled as a client/server pair. Different client/server pairs cannot establish usable sessions with each other by default.”

CVE CAN-1990-0660[22] loosely fits with the activity detected.

2.4.2 Reason This Detect Was Selected

- Backdoor Q is a very potent malware, definitely deserving a closer inspection.
- Backdoor rules file was commented out in the default Snort configuration file setup. This creates potential for Backdoor Q false negative.

2.4.3 Detect Was Generated By

The detect was triggered by following rule residing in backdoor.rules file:

```
alert tcp 255.255.255.0/24 any -> $HOME_NET any (msg:"BACKDOOR Q access"; dsize:>1;
flags:A+; flow:stateless; reference:arachnids,203; classtype:misc-activity; sid:184;
rev:6;)
```

The rule verbatim: Trigger on TCP traffic originating from any port on subnet 255.255.255.0/24, going to any port on home network. We specified home network in Snort command line –h option as 170.129.0.0/16. The TCP payload shall be greater than 1 byte. The TCP flag ACK has to be set, any other flag may be set. The rule

[20] Honeypot Project
[21] Gordon, Les.
[22] CVE [1].

ignores state of the TCP connection. Arachnids number 203 is provided as reference. We see signature classification as miscellaneous activity, signature identification 184 and signature revision 6.

How the triggering datagram correlates with the rule:

```
20:11:56.686507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4
(0x0800), length 60: IP (tos 0x0, ttl  14, id 0, offset 0, flags [none],
length: 43) 255.255.255.255.31337 > 170.129.166.76.515: R [tcp sum ok] 0:3(3)
ack 0 win 0 [RST cko]
        0x0000:  4500 002b 0000 0000 0e06 5c00 ffff ffff  E..+......\.....
        0x0010:  aa81 a64c 7a69 0203 0000 0000 0000 0000  ...Lzi..........
        0x0020:  5014 0000 1028 0000 636b 6f00 0000       P....(..cko...
```

The byte on offset 9th (0x06) matches TCP protocol. Source address on 12-15th bytes (0xffffffff or 255.255.255.255) fits in 255.255.255.0/24 range. Source port on offset 20-21 (0x7a69 or 31337) by default checks with “any” value. Destination address on offset 16-19 (0xaa818180 or 170.129.129.128) matches HOME_NET 170.129.0.0/16. Destination port on 22-23th byte (0x0203 or 515) again by default fits in “any” value. Ack flag is on, along with RST flag in byte 33 (0x14). Payload is 3 bytes on offset 40-42 (0x636b6f, ASCII “cko”). This value matches total datagram length in bytes 2-3 (0x2b or 43) minus IP header length (lower nibble in byte 0 with value 5, multiplied by coefficient 4 to the total of 20), minus TCP header length (upper nibble in byte 32 with value 5, multiplied by coefficient 4 to the total of 20). We have TRUE for 3 being greater than 1 (>1 is the rule). So all needed conditions in the rule are met to trigger on this datagram.

In the examined file there were 21 “Backdoor Q access” detects.

2.4.4 Probability the Source Address Was Spoofed

The source address was for sure spoofed. RFC 1122[23] describes in section 3.2.1.3 c) 255.255.255.255 as: “Limited broadcast. It MUST NOT be used as a source address. A datagram with this destination address will be received by every host on the connected physical network but will not be forwarded outside that network.” Source IP address protocol violation may be an attempt to evade IDS and/or true originator obfuscation. The source MAC address indicates extranet/Internet origin.

Supportive evidence: Source port 31337 (31337=eleet=elite in hacker slang) is as per Joakim von Braun[24] associated, as destination port, with many trojans: ADM worm, Back Fire, Back Orifice (Lm), Back Orifice russian, BlitzNet, BO client, BO Facil, BO2, Freak88, Freak2k, NoBackO. Destination port 515 (assigned by IANA[25] to printer spooler service) is not logical for legal traffic. TTL (always 14) is really low and inconsistent with the other log file traffic originated on Internet. IP Identification is always set to 0 – with real life probability of 2^{-16} of that happening in just two consecutive detect datagrams.

[23] IETF.
[24] Von Braun, Joakim.
[25] IANA.

2.4.5 Attack Mechanism

The qd daemon listens on the infected machine for traffic recognized as qs messenger no-response-required command. Then qd communication with commanding q client can take place. The Backdoor Q datagram fits in the qs command category.

Use of RST-ACK TCP flags combination can be rationalized as an IDS evasion attempt. It would also break any existing TCP connection. In the log there is no evidence of established TCP connection prior to the Backdoor Q packet, or any reply to this stimulus. But reply to IP address 255.255.255.255 would be limited to the local subnet anyway. In situation like this we would like to have sniffer logs of the wire traffic to investigate anomalies on Backdoor Q destination IP addresses in the period after receiving this datagram.

All above evidence points out that the datagram is a trigger of sorts. But the hypothetical association with Q has its weaknesses: The detect rule is really broad, with the source address being the only major filter. Use of destination port 515 may seem to be good due to its availability on most UNIX installations. But for example port 80 would be a better choice for penetrating the network filtering devices. And what to think about the noisy presentation: 21 hits against 21 unique targets, from 170.129.4.175 to 170.29.230.201? Not exactly stealth one-to-one relationship of Q. With such a low frequency it cannot be a DoS and a RST datagram does not persistently consume any local resources anyway. How about bad mutation of Q?

Based on Google search, Mark Stingley[26] claims the datagram to be generated by Sonicwall Integrated Security Appliance to clear a TCP connection violating an Intrusion Prevention Services policy. We could not reproduce Mark's search results. This scenario would take the detect datagram classification from a stimulus to a response.

The strongest hypothesis still remains that the datagram is scan to trigger trojan-infected hosts to act according to the trigger, possibly to contact the master.

2.4.6 Correlations

Already mentioned outstanding article by Les Gordon[27] examined Q behavior in laboratory environment. His traffic findings, leading to development of Q specific detect rules, are somehow different from datagram presented here. Our payload "cko" does not fit the clear text or encrypted payload Les recorded. But as Les points out: "The real danger is perhaps not from Q itself, but from other private software which makes use of similar "stealth" techniques which we don't have signatures for…" Les himself does not see correlation with the original Q.

Peter Storm[28], Rob McBee[29] and Sai Prasad Kesavamatham http://www.giac.org/practical/GCIA/SaiPrasad_Kesavamatham_GCIA.pdf agree with

[26] Stingley, Mark.
[27] Gordon, Les.
[28] Storm, Peter.
[29] McBee, Rob.

Les's conclusion that the packet is call-home request. They expanded on Q correlation with IRC. See other references in the detect text.

CVE CAN-1990-0660[30] loosely fits with the activity detected.

2.4.7 Evidence of Active Targeting

As per raw data available, none of the destination addresses had any prior reconnaissance targeted at them – there was even no other incoming or outgoing traffic then the event triggering datagrams. Number of the Backdoor Q events in files 2002.10.14 to 2002.10.18 were: 18, 32, 29, 43 and 21. This spread of the Backdoor Q access events in both IP and time dimensions just support our opinion that this is not a case of an active targeting.

2.4.8 Severity = 1

(2 + 5) – (3 + 3) = 1

Criticality	Unknown criticality of targets, but with high last IP address octet – less likely to be critical assets. No active targeting is a plus.	2
Lethality	Would a target be already infected, Q communication means root access and the game is over.	5
System Countermeasures	Unknown system level protection and patch management. No response is a plus.	3
Network Countermeasures	High probability of a firewall/filtering router on the internal subnet access point. Poor ingress filtering on the extranet access point. No response is a plus.	3

Table 5 Severity of BACKDOOR Q access

2.5 Detect 3 – CHAT MSN message

```
[**] [1:540:11] CHAT MSN message [**]
[Classification: Potential Corporate Privacy Violation] [Priority: 1]
11/18/02-05:41:53.946507 0:0:C:4:B2:33 -> 0:3:E3:D9:26:C0 type:0x800 len:0xD4
170.129.50.120:62174 -> 207.46.108.4:1863 TCP TTL:124 TOS:0x17 ID:59665 IpLen:20
DgmLen:198 DF
***AP*** Seq: 0x218B76  Ack: 0x312453C7  Win: 0x21E1  TcpLen: 32
TCP Options (3) => NOP NOP TS: 20420 3174379
```

2.5.1 Description of Detect

This detect, belonging to "acceptable use policy" group, triggers on client to server connection of MSNP – MSN Messenger Protocol. This is one of major Instant Messaging (IM) end user applications, also called chats. They are very popular, feature rich suites enabling exchange messages, files, hyperlinks and other objects on peer-to-peer basis. The detect means that an instance of MSN Messenger client was installed, configured and activated inside of the internal network.

[30] CVE [1].

One can argue that presence of IM datagrams in the log file demonstrates that ABC University IT Security department was concerned with Messenger and threats it poses. The IM threats can be clustered as information leak (by sending text in clear), worm access point, application vulnerability and threats related to direct peer-to-peer connectivity - IM circumvents the perimeter protection.

Examples of related MSN CVE numbers: CAN-2002-0155[31] (Buffer overflow allows remote arbitrary file execution), CAN-2002-0472[32] (Weak authentication allows remote spoofing of messages from other users), CAN-2004-007[33] (Buffer overflow allows remote denial of service and possibility of arbitrary code execution), CAN-2004-0122[34] (Information leak vulnerability – remote read of arbitrary files).

2.5.2 Reason This Detect Was Selected

- Not a single chat related detect was generated with the default Snort rule set. Once the rules were adjusted as per section 3.2, (all) 169 egress datagrams to destination port 1863 traffic was identified as MSN IM.
- Instant Messaging poses significant security risk, not quite understood by an average IM user.

2.5.3 Detect Was Generated By

This detect was triggered by following rule residing in chat.rules file:

```
alert tcp $HOME_NET any <> $EXTERNAL_NET 1863 (msg:"CHAT MSN message";
flow:established; content:"MSG "; depth:4; content:"Content-Type|3A|";
nocase; content:"text/plain"; distance:1; classtype:policy-violation;
sid:540; rev:11;)
```

The rule description: Raise an alert on TCP traffic originating from any port on HOME_NET, specified in Snort command line –h option as 170.129.0.0/16, going to port 1863 on external network (set to any IP address in snort.conf). The banner message for this rule is “CHAT MSN message”. Rule identification number is 540, revision 11. We see the signature classification type as policy violation.

The rule specifies that there must be several payload elements present in the datagram. The first is string “MSG “, case sensitive, to be located within first 4 bytes of the payload, effectively on the beginning of it. The second one is the string “Content-Type” followed by value 0x3A, anywhere in the payload, case not sensitive. The last one is “text/plain”, case sensitive, to be matched starting one byte from the end of previous match.

The most interesting part of the rule is “flow:established” option. It specifies TCP connection to be established, as determined by preprocessor stream4. See section 3.2 Snort for details. Because in our configuration stream4 is disabled, the option reverts to

[31] CVE [3].
[32] CVE [4].
[33] CVE [5].
[34] CVE [6].

the previous implementation “flags A+” (TCP flag ACK to be present, other flags may be present). To prove stream4 non-suitability for our log with only detects related datagrams, we temporarily enabled stream4. As a result all “CHAT MSN message” detects disappeared, along with other 18 detects.

How the triggering datagram correlates with this rule:

```
05:41:53.946507 00:00:0c:04:b2:33 > 00:03:e3:d9:26:c0, ethertype IPv4 (0x0800), length
212: IP (tos 0x17,CE, ttl 124, id 59665, offset 0, flags [DF], length: 198)
170.129.50.120.62174 > 207.46.108.4.1863: P [tcp sum ok] 2198390:2198536(146) ack
824464327 win 8673 <nop,nop,timestamp 20420 3174379>
        0x0000:  4517 00c6 e911 4000 7c06 fcdc aa81 3278  E.....@.|.....2x
        0x0010:  cf2e 6c04 f2de 0747 0021 8b76 3124 53c7  ..l....G.!.v1$S.
        0x0020:  8018 21e1 b1d0 0000 0101 080a 0000 4fc4  ..!...........O.
        0x0030:  0030 6feb 4d53 4720 3420 4e20 3133 330d  .0o.MSG.4.N.133.
        0x0040:  0a4d 494d 452d 5665 7273 696f 6e3a 2031  .MIME-Version:.1
        0x0050:  2e30 0d0a 436f 6e74 656e 742d 5479 7065  .0..Content-Type
        0x0060:  3a20 7465 7874 2f70 6c61 696e 3b20 6368  :.text/plain;.ch
        0x0070:  6172 7365 743d 5554 462d 380d 0a58 2d4d  arset=UTF-8..X-M
        0x0080:  4d53 2d49 4d2d 466f 726d 6174 3a20 464e  MS-IM-Format:.FN
        0x0090:  3d4d 5325 3230 5361 6e73 2532 3053 6572  =MS%20Sans%20Ser
        0x00a0:  6966 3b20 4546 3d42 3b20 434f 3d66 6630  if;.EF=B;.CO=ff0
        0x00b0:  3066 663b 2043 533d 303b 2050 463d 3232  0ff;.CS=0;.PF=22
        0x00c0:  0d0a 0d0a 6869                           ....hi
```

The byte on offset 9th (0x06) matches TCP protocol. Source address on offset 12-15 (0xaa813278 or 170.129.50.120) is covered by 170.129.0.0/16. Source port fits to “any” specification by default, the same applies to the destination address. The destination port on offset 22-23 (0x747 or 1863) matches. TCP ACK and PSH flag are on in byte 33 (0x18), indicating established TCP connection (absence of SYN, FIN or RST flags) – fits to the non-stream4 definition of established TCP connection.

The TCP payload starts at offset 52: IP header length (lower nibble in byte 0 with value 5, multiplied by coefficient 4 to the total of 20), plus TCP header length (upper nibble in byte 32 with value 8, multiplied by coefficient 4 to the total of 32). The first content string “MSG “ starts on offset 52 (0x4d534720), satisfying both case sensitivity and comparison depth of 4 bytes from beginning of TCP payload. We can see “Content-Type:” at the offset 84, matching the second string. One byte further there is “text/plain” matching with case sensitivity. All conditions of the rule are met.

Note: The IP Type of Service (ToS) field has a curious value 0x17. It is interpreted as follows: both ECN Capable and CE (Congestion Experienced) bits are set, along with Differentiated Services code 0x05, unknown to ethereal decoder. TCP connection setup would be needed to examine legality of this value. The sparse traffic pattern does not suggest that any network congestion is taking place. MSN detect by Rob Perdue[35] shows standard ToS value 0x0.

[35] Perdue, Rob.

2.5.4 Probability the Source Address Was Spoofed

The source address is probably not spoofed. “CHAT MSN message” detects are triggered on sequence of instant messages sent with MSN Messenger in an established TCP connection. It would be impossible to keep timing and acknowledgements correct in spoofed scenario for 69 minutes of the TCP connection log time, without breaking the connection. From the application perspective: the TCP payload shows (one way) continuity of the discussion at hand. In case of spoofed address the attacker would not see the responses (unless sniffed along the way) and could not continue a meaningful conversation.

Additional point: The whole session has consistent TTL value 124.

2.5.5 Attack Mechanism

Snort illustrates possible associated attack scenario in description of the above rule: “An attacker might utilize vulnerability in an IM client to gain access to a host, then upload a Trojan Horse program to gain control of that host.” Here is a quick run on exposure groups in IM:

Information Leak

Just scrolling through the datagrams in this log reveals discussion of user’s household events. The problem is that the conversation is communicated in clear text, give-away to a sniffer running along the communication path. Information leaks may be also result of vulnerability, such as disclosure of the user’s name and email address reported by Bugtraq Id 4028[36].

Worm Access Point

viruslist.com[37] lists these MSN Messenger-specific worms: I-Wrom.Choke.a, I-Worm.Newpic.a and MSN-Worm.Jitux. Especially dangerous aspect of IM based worms is speed of their propagation. According to recent article by Gregg Keizer[38]: “… an IM worm along the lines of MSBlast or Slammer could simply hijack the contact lists of vulnerable clients, then use that list to send itself to others... half a million systems could be infected in a little as 30 to 40 seconds.”

Open Vulnerabilities

Paul Robers[39] reports: “Currently, there are about 60 published IM vulnerabilities…” Some of them are listed with their CVE number in 2.5.6 Correlations.

2.5.6 Correlations

David Merkle explains how Snort TCP stream preprocessor (stream4) would disable “CHAT MSN message” detects in http://isc.sans.org/logs/Raw/ logs: http://cert.uni-stuttgart.de/archive/intrusions/2003/11/msg00013.html.

[36] SecurityFocus [2].
[37] viruslist.com
[38] Keizer, Gregg.
[39] Roberts. Paul.

Mike Mintz presents example of the MSN Messenger session, shows TCP port 1863 as client to server vector:
http://www.hypothetic.org/docs/msn/notification/example_session.php.
Rob Perdue[40] examines different groups of security concerns related to IM.

Examples of related MSN CVE numbers: CAN-2002-0155[41], CAN-2002-0472[42], CAN-2004-007[43] and CAN-2004-0122[44].

2.5.7 Evidence of Active Targeting

There is no evidence of active targeting. Based on the continuity of conversation thread in the log datagram sequence we can assert that the examined datagram is part of IM conversation between two MSN Messenger clients. Event count in the context files 2002.10.14 to 2002.10.17: 0, 14, 0, 0 suggests low frequency of IM use in the ABC University network.

2.5.8 Severity = 3

(2+ 5) – (2+ 2) = 3

Criticality	Unknown type of system running MSN Messenger client. Relationship nature of discussion presented in the payload indicates a "common user" host. High last IP address octet suggests non-critical asset. Also MSN Messenger is not likely to be installed on a network-critical host.	2
Lethality	The datagram is not an attack. But there are many vectors and exploits targeting MSN Messenger. The worse case scenarios listed above include arbitrary code execution, with possible impact of remote Administrator access and control.	5
System Countermeasures	There is no evidence of MSN Messenger patch level, antivirus or firewall running on the host.	2
Network Countermeasures	The traffic is allowed through the internal network perimeter protection, which is present with high probability.	2

Table 6 Severity of CHAT MSN message

2.6 Network Statistics

For methodology and tools used see 3.5 Network Statistics. As described there, weighting factor is product of detect directionality and perceived severity. For example

[40] Perdue, Rob.
[41] CVE [3].
[42] CVE [4].
[43] CVE [5].
[44] CVE [6].

"CHAT MSN message" is as egress traffic assigned factor 4, although more complex severity weighting methodology used in section 2.5.8 results in its overall severity of 3.

2.6.1 Top Five Talkers

Position	IP Address	Mostly Used Dest. Port #	Count
1	170.129.50.120	TCP 80/WWW	1364
2	202.108.254.204	TCP 3128/SQUID TCP 8080/WWW, Webcache	47
3	63.111.48.133	80/WWW	28
4	255.255.255.255	TCP 515/Printer Spooler	21
5-6	211.47.255.23	TCP 0/Reserved	16
5-6	153.33.24.3	UDP 111/Sun RPC Portmapper	16

Table 7 Top Five Talkers by Datagrams Generated

Detect Count Position	IP Address	Total Detect Count	Event Name	Event Count	Traffic Direction	Weight Factor	Weighted Count - Detect	Total Weighted Count - IP	Final Position
1	170.129.50.120	169	CHAT MSN message	169	egress	4	676	676	1
2	202.108.254.204	47	SCAN SQUID Proxy attempt	16	ingress	1	16		2
			SCAN Proxy Port 8080 attempt	16	ingress	1	16		
			SCAN SOCKS Proxy Attempt	15	ingress	1	15	47	
3	255.255.255.255	21	BACKDOOR Q access	21	ingress	2	42	42	3
4-5	153.33.24.3	16	RPC portmap mountd request UDP	16	ingress	2	32	32	4
4-5	211.47.255.23	16	BAD-TRAFFIC tcp port 0 traffic	16	ingress	1	16	16	5

Table 8 Weighted Top Five Talkers – Detects Only

The weighting methodology had some impact on the list. Egress traffic kept leading position despite triggering only on MSN Messenger detects. IP address 63.111.48.133 disappeared as not triggering any detect. Order in position 4-5 got clarification with increased weight of "RPC portmap mountd request UDP".

2.6.2 Top Targeted Ports

Position	Port	Service	Count
1	80	HTTP	1280
2	1863	MNSP MSN Messenger Client-To-Server	169
3	515	Unix Printer Spooler	21
4-7	8080	HTTP (proxy)	16
4-7	3128	Active API Server Port (Squid Proxy)	16
4-7	111	SUN Remote Procedure Call	16
4-7	0	Reserved	16

Table 9 Top Five Targeted Ports

Detect Count Position	Port	Total Detect Count	Event Name	Traffic Direction	Weight Factor	Weighted Count - Detect	Final Position
1	1863	169	CHAT MSN message	egress	4	676	1
2	515	21	BACKDOOR Q access	ingress	2	42	2
3-6	111	16	RPC portmap mountd request UDP	ingress	2	32	3
3-6	8080	16	SCAN Proxy Port 8080 attempt	ingress	1	16	4-6
3-6	3128	16	SCAN Squid Proxy attempt	ingress	1	16	4-6
3-6	0	16	BAD-TRAFFIC tcp port 0 traffic	ingress	1	16	4-6

Table 10 Weighted Top Five Ports – Detects Only

Impact of the weighting methodology was basically identical as to the top talkers. Egress traffic to Web servers vanished – no detects associated with it. The rest of top ports got slightly reordered due to factoring in the detect significance.

2.6.3 Three Most Suspicious External Source IP Addresses

61.140.72.65

We analyzed this IP as case of possible active targeting on right platform in 2.3 Detect 1 - WEB-FRONTPAGE _vti_rpc access.

```
inetnum:      61.140.72.64 - 61.140.72.79
netname:      GUANGZHOU-ELEC-COMM-CENTER
descr:        GUANGZHOU ELECTRIC COMMUNICATION CENTER
country:      CN
…
```

TTL value of 45 suggests initial TTL equal 64, resulting in (a higher, but still reasonable) 19 hops distance to the originator. Value 64 is associated with p0f OS estimate: many Linux and one Windows 9x entry. Window size of 5840 narrowed it down to a few choices in the Linux 2.4.1-2.4.14 range.

64.105.70.231

This IP triggered a single FrontPage detect directed at internal Web server running FPSE. Such scenario may be of concern on an unpatched server.

```
OrgName:    Covad Communications
OrgID:      CVAD
Address:    2510 Zanker Rd
City:       San Jose
StateProv:  CA
…
```

TTL value of 113 suggests 15 hops from the initial value128, associated in p0f with Windows. Window size 64240 matched one entry only: Windows XP Pro, Widows 2000 Pro.

202.108.254.204

This is the noisiest external IP address. With none of external sources being too threatening, we may wish to learn more about this “loud one”.

```
inetnum:      202.108.0.0 - 202.108.255.255
netname:      CNCGROUP-BJ
descr:        CNCGROUP Beijing province network
descr:        China Network Communications Group Corporation
descr:        No.156 Fu-Xing-Men-Nei Street
descr:        Beijing 100031
country:      CN
…
```

Running p0f on the log file identifies this IP as "2002.108.254.204 [19/20 hops]: NMAP scan (distance inaccurate) (1)". Indeed, according to nmap man file[45] the option –ttl "Sets the Ipv4 time to live field in sent packets to the given value." All attempts to qualify the OS, based on TTL, are then off. Still TTL value 45 and 46 are 18 to 19 hops (reasonable value) from 64, mostly associated byp0f with UNIX/Linux/Macintosh. There is a good chance that the source IP is running UNIX flavor without TTL modification.

2.7 CGIA and Other Correlations

See correlations and references throughout the text.

2.7.1 Correlations with Current Trends from Internet Storm Center

The data log file is almost two years old, so we should keep that in mind while correlating with current state of Internet affairs. With no malicious traffic leaving the internal network (see section 2.2.3) we used the ingress destination port profile in Table 1 as log data reference. Present mal-activity snapshot was taken from SANS Internet Storm Center (ISC) Top 10 page: http://isc.sans.org/top10.php on September 5, 2004. Common grounds with ISC top 10 ports:

Port	Datagram Count	Source IP Address	Detect Count
80	85	Various	17
139	6	Various	0
1080	15	202.108.254.204	15

Table 11 Log Correlation with SANS ISC Top 10 ports

Times change, but exploits against Web servers, Microsoft networking and SOCKS proxies stay. There is no correlation with current ISC top 10 IP address list.

2.8 Internal Network Compromise, Dangerous or Anomalous Activity

As summarized in sections 2.2.2 and 2.2.3, the network does not exhibit any symptoms of compromise or anomalous egress traffic. The only area of concern is use of MSN Instant Messaging, as discussed in section 2.5.

[45] insecure.com.

2.9 Defensive Recommendations

- Setup perimeter firewall and, if not already present, another one on internal network access point. Basic rule: deny what is not allowed, even in DMZ. All ingress traffic should be allowed for only traffic originated from inside. As needed, allow inbound traffic to public servers. Use deep packet inspection for application level filtering to control chat and other potentially tunneled applications.
- Log Cisco-2 activity. NAT logs are instrumental for identifying the real IP address of events associated with 170.129.50.120.
- The logs above should be made available for real time and forensics examination. An example of suitable technology: Log collection, analysis and retention appliances by LogLogic[46].
- Setup archive of traffic dump logs on network access points for correlation with other collected data.
- Setup host based IDS system on critical servers and NIDS on internal network to protect internal assets.
- To correlate all above data, time synchronization is essential on all logging devices.
- Keep all networked devices patches up to date.
- Ensure that Web servers are protected by an additional layer of defense. Solution examples would be InterDo[47], a Web application firewall by Kavado or application proxy.
- Define and enforce minimal host configuration requirements to run IM client.
- Eric Chien quoted in article by Gregg Keizer[48]:

  > Implement IM usage policy that lays out what public IM clients are permitted and how they can be used. If possible, steer clear of public IM networks and instead use an enterprise-level IM clients that sit inside the firewall for intra-company communication.

In Snort tuning process, ensure monitoring of real assets while minimizing the false positives: The internal network infrastructure should be mapped to Snort configuration. General, non-specific rules with high level of false positives or rules that are not relevant to permitted usage policy should be removed. Lastly site specific rules should be modified or created to fit the specific monitored hosts.

3 Analysis Process

We decoupled findings from the analytical procedures, as requested in the Practical requirement. We strived to minimize disconnect this may have caused. As warranted by need for document flow, minor procedural elements are specified in the section 2 Detailed Analysis.

[46] LogLogic.
[47] Kavado.
[48] Keizer, Gregg.

3.1 Processing Environment

The log files had been processed in the following environment:

- PC P4-M, 256 MB RAM, running RedHat 9
- Snort Version 2.2.0RC[49], with default rule set version 2.1.0 (dated June 3, 2004) with modifications as below.
- SnortSnarf[50] Version 2.1111.1, cofigured as per alt.don[51]
- libpcap Version 0.8.3
- tcpdump[52] Version 3.8.3
- p0f[53] Version 1.8.3

3.2 Snort Tuning

The latest Snort rules version 1.142.2.2[54] dated August 5, 2004 were downloaded and tested. It resulted in only less than half of the alarms delivered with default rules from Snort version 2.2.0RC1. So the default rule set from the Snort release was used.

During the snort.conf file review the backdoor.rules rule file was added to the default configuration (by un-commenting in snort.conf). Watching for backdoors is, based on our experience, important in any monitored environment.

Daniel Wesemann[55] recommends to turn off, for the sake of recovering maximum of the original detects, following Snort preprocessors:

```
# preprocessor stream4: detect_scans, disable_evasion_alerts
# preprocessor stream4_reassemble
```

As we found out, the stream4 preprocessor keeps state of the TCP connections as established in SYN, SYN-ACK, ACK datagrams. The examined log file contains only the datagrams triggering detects, not the original TCP session setup datagrams. Such scenario renders stream4 counterproductive, because the preprocessor filters out the detect datagrams as not being part of established TCP connections.

At this point 73% detects were from http_inspect group, targeting external servers. Review of snort.conf and README.http_inspect showed that by default the server address is “any”. As determined in the section 2.2.2 Web Server, the only Web server in internal network has IP address 170.129.50.3. Given the Apache and FPSE environment, we used “all” for server (type) profile, giving Snort wide application level target. Once we implemented following inspection focus, http_inspect detect count dropped to just three:

```
var HTTP_SERVERS 170.129.50.3
preprocessor http_inspect_server: server 170.129.50.3 \
    profile all ports { 80 } oversize_dir_length 500
```

[49] Snort [1].
[50] SnortSnarf.
[51] alt.don.
[52] tcpdump.
[53] Zalewski, Michal [1].
[54] Snort [2].
[55] Wesermann, Daniel.

We were still puzzled by low detect-to-datagram ratio in the 2002.10.18 log file. Scrolling through the file I noticed significant outbound traffic to TCP port 1863 MSNP[56] - do we have IM chat on MSN Messenger? Our load of snort.conf has by default chat rules disabled. Once enabled, number of detects jumped from 143 to 308:

```
include $RULE_PATH/chat.rules
```

3.3 Network Topology

3.3.1 Description

MAC addresses, tightly coupled with the underlying networking hardware, are very reliable vehicle to identify true point-to-point and end-to-end relationships. Peter Storm[57] used tcpdump and assorted tools to MAC to IP correlation. Kah-Leong Fong[58] employed similar method, with different sorting tools.

First the source and destination L2 addresses are enumerated. That establishes the neighborhood point-to-point topology of the Snort monitoring point. Next the source and destination IP addresses are enumerated and counted for every source L2 address, revealing associated L3 connection end points. In the end we can see what IP address range is behind every L2 point, and at the far side of the connection. Destination ports are also enumerated to reveal what ingress and egress services are allowed at the monitoring point.

We use method by Rob Perdue[59], where the variations in egress Web field User-Agent and IP field TTL (time to live) indicate presence of multiple hosts and routers behind a single IP address.

Finally the collected data are evaluated. Anomalies such as of asymmetry of traffic on IP or service basis, or spoofing of internal addresses are very pronounced. We have enough data to draw a network diagram.

Word of caution: The network analysis is to a degree tilted by log file containing only the packets matching Snort rules and also by datagram removal due to content sanitization.

3.3.2 Toolkit

To enumerate L2 source and destination points:

```
tcpdump -ner 2002.10.18 | awk '{print $2}' | sort –u
tcpdump -ner 2002.10.18 | awk '{print $4}' | sort –u
```

Where:

tcpdump:

`-n`	Don't convert addresses to names.

[56] IANA.
[57] Storm, Peter.
[58] Fong, Kah-Leong.
[59] Perdue, Rob.

```
      -e            Print link-level header on each dump line.
      -r 2002.10.18
                    Read data from file 2002.10.18.
awk:
      '{print $2}'  Print only L2 source field.
      -F \.         Use "." for the input field separator.
sort:
      -u            Output only first of equal inputs.
```

One source L2 address perspective: L3 source and destination address enumeration and counting, destination port enumeration and counting:

```
tcpdump -ner 2002.10.18 ether src 0:3:e3:d9:26:c0 | awk '{print $11}' | awk -F \.
'{print $1 "." $2 "." $3 "." $4}' | sort –u
tcpdump -ner 2002.10.18 ether src 0:3:e3:d9:26:c0 | awk '{print $11}' | awk -F \.
'{print $1 "." $2 "." $3 "." $4}' | sort –u | wc -l
tcpdump -ner 2002.10.18 ether src 0:3:e3:d9:26:c0 | awk '{print $13}' | awk -
F \. '{print $1 "." $2 "." $3 "." $4}' | awk -F \: '{print $1}' | sort –u
tcpdump -ner 2002.10.18 ether src 0:3:e3:d9:26:c0 | awk '{print $13}' | awk -
F \. '{print $1 "." $2 "." $3 "." $4}' | awk -F \: '{print $1}' | sort –u | wc
-l
tcpdump -ner 2002.10.18 ether src  0:3:e3:d9:26:c0 | awk '{print $13}' | awk
-F \. '{print $5}' | sort –u | sort –n
tcpdump -ner 2002.10.18 ether src  0:3:e3:d9:26:c0 | awk '{print $13}' | awk
-F \. '{print $5}' | sort –u | sort –n | wc -l
```

Where:

```
tcpdump:
      ether src 00:03:e3:d9:26:c0
                    Show only traffic with source Ethernet MAC address
                    00:03:e9:26:c0.
wc:
      -l            Count lines.
sort:
      -n            Compare according to string numerical value.
```

Verification that there are no datagram with the same source and destination MAC address:

```
tcpdump -ner 2002.10.18 'ether src 0:0:c:4:b2:33 and ether dst 0:0:c:4:b2:33'
tcpdump -ner 2002.10.18 'ether src 00:03:e3:d9:26:c0 and ether dst 00:03:e3:d9:26:c0'
```

Example of Ethereal filter to separate Web traffic for User-Agent and TTL analysis:

```
(ip.src == 170.129.50.120 and tcp.dstport == 80 and http.request)
```

3.3.3 Missing Detects

To find all log file datagrams that do not generate detects, we inverted snort alerting algorithm:

- cat-ed all included rule files to snort.conf, removed the "include" statements
- Using vi replaced "alert" keyword with "pass" in all rules:
  `:%s/alert/pass/gc`
- Replaced back "nopass" to "noalert" keyword (collateral damage of global replacement)

- Changed the order of Snort action to Pass->Alert->Log

The binary file containing no-detect datagrams was generated with:

```
snort –c ../snort-2.2.0RC1/etc/snort.test -r 2002.10.18  -k none -A full -l
../log/2002.10.18.test -o -bL ./test
```

Where:

`-o` Change rules processing order to Pass->Alert->Log

`-bL ./test` Write in binary mode to file test (with appended timestamp)

3.4 Detect Generation and Analysis

Detects were generated using the following command line:

```
snort -c ../snort-2.2.0RC1/etc/snort.conf -r 2002.10.18 -h 170.129.0.0/16 -l
../log/2002.10.18 -k none -A full -dyev > ../../log/2002.10.18/verbose
```

Where:

`-c ../snort-2.2.0RC1/etc/snort.conf`
Run in intrusion detection mode using rule files specified in snort.conf

`-r 2002.10.18`
Read and process from the file 2002.10.18

`-h 170.129.0.0/16`
Set HOME_NET variable to 170.129.0.0/16

`-k none`
Turn off checksum verification.

`-A full` Use full alert mode. The alert file contains full decoded header as well as the alert message.

`-d` Dump the application layer.

`-y` Include year in alert and log files

`-e` Display the link layer packet header

`-v` Output verbose to console. Redirected to ../../log/2002.110.18/verbose

Once Snort output had been generated, the signatures had been sorted by SnortSnarf:

```
./snortsnarf.pl -rs -d ../html/2002.10.18 ../log/2002.10.18/alert
```

Where:

`-rs` Reverse signature listing order, put most interesting first.

`-d ../html/2002.10.18`
Set output directory to ../html/2002.10.18.

`…/log/2002.10.18/alert`
Text file containing Snort detects.

tpcdump was used for packet decode presentation. Following example is set up for Backdoor Q detect packets:

```
tcpdump -neXvvvSs 0 -r 2002.10.18 'src host 255.255.255.255'
```

Where (only option not listed above):

`-X` Print each packet (minus its link level header) in hex and ASCII.

`-vvv` Most verbose output.

`-S` Print absolute, rather than relative, TCP sequence numbers.

`-s 0` Capture all packet data available.

`'src host 255.255.255.255'`

Capture only data with source IP address 255.255.255.255.

At times ethereal provides better view of situation. Example of the filter used:

```
ip.src == 170.129.50.3 and ip.dst == 64.154.80.45
```

3.5 Network Statistics

3.5.1 Top Talkers

To present the overall network traffic dynamics, we first present top five IP address sorted solely by number of sent datagrams.

We feel that there are several important parameters that signify traffic producers:

- Datagrams associated with detect(s) are more significant than other traffic
- Egress detects are much more important than ingress ones
- Detects differ by severity

Recent heuristic detect correlation tool simtransd[60] from Enterasys puts also emphasis on egress events and their sequencing, while discounting the ingress events. So we devised a second top talker list based solely on the Snort alert file. The outbound detects have additional weight multiplication factor of 4. Detects judged in our discretion to be severe have additional weight multiplication factor of 2.

3.5.2 Top Targeted Ports

Top targeted ports were approached by the same strategy as the top talkers. First we sorted the whole log file by descending count of datagram destination ports of the same number. For the second list we included only datagrams with detects and applied the weighting factor of direction and detect severity.

3.5.3 Toolkit for Top Talkers and Targeted Ports

In addition to already listed sources, we draw tool ideas from Ian Martin[61].
To generate list of IP addresses sorted by number of datagrams originated:

```
tcpdump -nr 2002.10.18 | awk '{print $3}' | awk -F \. '{print $1 "." $2 "." $3 "."
$4}' | sort | uniq -c | sort –rn
```

To generate Snort alert as one line per detect, which facilitates easy parsing:

```
snort -c ../snort-2.2.0RC1/etc/snort.conf -r 2002.10.18 -h 170.129.0.0/16 -l
../log/2002.10.18.fast -k none -A fast
```

File modification: “fast” Snort alarm file contained “(http_inspect) NONRFC HTTP DELIMITER” detect, that had two standard fields missing. The fields were added to produce consistent parsing results.

To count events per source IP address:

[60] Golomb, Gary.
[61] Martin, Ian.

```
grep 255.255.255.255 alert | sed 's/\[\*\*\]//g' | awk -F ] '{print $2}' | awk -F [
'{print $1}' | sort | uniq -c | sort -r
```

To generate descending list of destination port by occurrence:

```
tcpdump -nr 2002.10.18 | awk '{print $5}' | awk -F \. '{print $5}' | sort | uniq -c |
sort -rn
```

During port sorting we saw 20 occurrences of empty port field, caused by datagram having corrupt TCP section. tcpdump ignored port decode in this situation.

To list count of ports and associated detects:

```
sed 's/\[\*\*\]/:/g' alert |  awk -F \: '{print $11 ":  " $6}' | awk -F \] '{print  $1
":" $2}' | awk -F \: '{print $1 $3}' | sort | uniq -c | sort -rn
```

3.5.4 Three Most Suspicious External Sources

As IP address registration search starting point we used www.name-space.org, then followed to the regional registration authority as needed. www.samspade.org was used as alternative registration search engine.

For the examined IP addresses in most cases we do not have available SYN or RST packet, used for building p0f passive fingerprinting database. We still utilized this tool with established TCP connection datagrams, although p0f author Michal Zalewski[62] warns: “It is possible to perform passive fingerprinting on a live TCP connection… However, these techniques are less reliable…” Parameters TTL and window size were used for manual OS triangulation from p0f database file p0f.fp. The actual datagram size was contemplated as another assessment point, but the log datagrams were too short to provide any OS differentiation.

To process log file with p0f, resulting in OS assessment for only IP addresses with TCP connection setup and RST datagrams:

```
p0f –s 2002.10.18
```

[62] Zalewski, Michal [2].

4 Appendix

4.1 List of References

alt.don. “SnortSnarf tutorial for Linux.” Jul 23, 2003. URL: http://www.security-forums.com/forum/viewtopic.php?p=104305 (Jun 8, 2004).

CVE [1]. “CAN-1999-0660 (under review).” URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=can-1999-0660 (Aug 13, 2004).

CVE [2]. “CAN-2001-0096.” URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=2001-0096 (Aug 20, 2004).

CVE [3]. “CVE-2002-0155.” URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0155 (Aug 29, 2004).

CVE [4]. “CAN-2002-0472.” URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0472 (Aug 29, 2004).

CVE [5]. “CAN-2004-0007.” URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0007 (Aug 29, 2004).

CVE [6]. “CAN-2004-0122.” URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0122 (Aug 29, 2004).

Embyte. “Gspoof.” Mar 10, 2004. URL: http://gspoof.sourceforge.net/ (Sep 5, 2004).

Fong, Kah-Leong. “GCIA Certified Intrusion Analyst (GCIA) GCIA Pactical Assignments, version 3.3.” Aug 4, 2003. URL: http://www.giac.org/practical/GCIA/Kahleong_Fong_GCIA.pdf (Aug 5, 2004).

Golomb, Gary. “RE:[Dragonidsuser] Event Correlation with simstransd.” Email to Dragon IDS Mailing List. Aug 20, 2004. URL: https://dragon.enterasys.com/modules.php?op=modload&name=PostWrap&file=index&page=support

Gordon, Les. “On Q” 2004. URL: http://www.whitehats.ca/main/publications/external_pubs/Q-analysis/Q-analysis.html (Aug 13, 2004).

Hall, Eric A. Internet Core Protocols. O’Reilly. Feb 2000. p.159.

Honeypot Project. “Attacker tools found on apollo.honeyp.edu.” URL: http://www.honeynet.org/challenge/results/submissions/addam/toolkit.txt (Aug 13, 2004).

Hutley, Brett. “LOGS: GIAC GCIA Version 3.4 Practical Detect Brett Hutley.”

Sep 1, 2004. Email from brett@hutley.net to mailgroup intrusions@lists.sans.org. URL: http://lists.sans.org/pipermail/intrusions/

IANA. "PORT NUMBERS" Aug 13, 2004. URL: http://www.iana.org/assignments/port-numbers (Aug 14, 2004).

IEEE. http://standards.ieee.org/regauth/oui/oui.txt. No date. URL: http://standards.ieee.org/regauth/oui/oui.txt (Aug 10, 2004).

IETF. "RFC 1122 - Requirements for Internet Hosts - Communication Layers." Oct 1989. URL: http://www.faqs.org/rfcs/rfc1122.html (August 14, 2004).

insecure.com: "Nmap network security man page." 2004. URL: http://www.insecure.org/nmap/data/nmap_manpage.html (Sep 5, 2004).

Jetmore, John [1]. "possible frontpage exploit?" Jul 16, 2001. URL: http://lists.jammed.com/incidents/2001/07/0067.html (Aug 20, 2004).

Jetmore, John [2]. "Re: possible frontpage exploit?" Jul 16, 2001. URL: http://lists.jammed.com/incidents/2001/07/0074.html (Aug 20, 2004).

Kavado. "InterDo Web Application Firewall" 2004. URL: http://www.kavado.com/products/interdo.asp (Aug 23, 2004).

Keizer, Gregg. "IM Worms Could Spread In Seconds." Jun 21, 2004. URL: http://www.internetweek.com/breakingNews/showArticle.jhtml%3Bjsessionid=YKE02ZGNN21GKQSNDBCSKHY?articleID=22101033 (Aug 29, 2004).

Lampe, John. "IIS FrontPage DoS Script" 2000. URL: http://cvsweb.nessus.org/cgi-bin/cvsweb.cgi/~checkout~/nessus-plugins/scripts/IIS_frontpage_DOS_2.nasl?content-type=text/plain (Aug 22, 2004).

LogLogic. "LogLogic – Your Tool For Log Lifecycle Management". 2004. URL: http://www.loglogic.com/products/ (Sep 8, 2004).

Martin, Ian. "SANS Practical Version 3.3" July 17, 2003. URL: http://www.giac.org/practical/GCIA/Ian_Martin_GCIA.pdf (Sep 4, 2004).

McBee, Rob. "GCIA Certified Intrusion Analyst (GCIA) GCIA Pactical Assignments, version 3.3." Jul 8, 2003. URL: http://www.giac.org/practical/GCIA/Rob_McBee_GCIA.pdf (Aug 10, 2004).

Microsoft [1]. "Microsoft Frontpage 2000 Server Extensions SR1.2: Downloads for UNIX-Based Servers." Sep, 2002. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnservext/html/Unixfpse.asp (August 21, 2004).

Microsoft [2]. “Microsoft FrontPage Server Extensions 2002 for UNIX.” Sep 2002. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnservext/html/sr14.asp (Aug 20, 2004).

Perdue, Rob. “LOGS: GIAC GCIA Version 3.5 Practical Detect by Rob Perdue.” Aug 13, 2004. Email to mailgroup intrusions@lists.sans.org. URL: http://lists.sans.org/pipermail/intrusions/

Roberts, Paul. “IM Worms Pose Growing Threats.” Sep 26, 2003. URL: http://www.pcworld.com/news/article/0,aid,112667,00.asp (Aug 29, 2004).

SecurityFocus [1]. “Microsoft IIS Front Page Server Exensions DoS Vulnerability.” 2004 URL: “http://www.securityfocus.com/bid/2144 (Aug 22, 2004).

SecurityFocus [2]. “Microsoft MSN Active X Object Information Disclosure Vulnerability.” Feb 11, 2002. URL: http://www.securityfocus.com/bid/4028/info/ (August 29, 2004).

Snort [1]. “snort-2.2.0RC1.tar.gz” Version 2.2.0RC1. Jun 29, 2004. URL: http://www.snort.org/dl (Aug 10, 2004).

Snort [2]. “snortrules-snapshot-CURRENT.tar.gz” Version 1.42.2.2. Aug 10, 2004. URL: http://www.snort.org/dl/rules (Aug 10, 2004).

Snort [3]. “Snort Signature Database SID 937.” URL http://www.snort.org/snort-db/sid.html?id=937 (Aug 22, 2004).

SnortSnarf. URL: http://www.silicondefense.com/software/snortsnarf Note: As of Aug 10, 2004 this site was not available.

Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Addison-Wesley, 2003.

Stingley, Mark. “GIAC GCIA Version 3.5 Practical Detect Mark Stingley.” September 18, 2004. Email from cw3sting@yahoo.com to mailgroup intrusions@lists.sans.org. URL: http://lists.sans.org/pipermail/intrusions/

Storm, Peter. “GIAC Certified Intrusion Analyst (GCIA) Practical Assignment.” Nov 15, 2003. URL: http://www.giac.org/practical/GCIA/Pete_Storm_GCIA.pdf (August 5, 2004).

tcpdump. “TCPDUMP public repository.” Version 3.8.3. URL: http://www.tcpdump.org (May 9, 2004).

Viruslist.com: “Virus Encyclopedia, Internet Worms, Messenger.” Aug 30, 2004. URL: http://www.viruslist.com/eng/viruslist.html?id=4240 (Aug 30, 2004).

Von Braun, Joakim. “The Trojan List.” Oct 15, 2002. URL: http://www.simovits.com/sve/nyhetsarkiv/1999/nyheter9902.html (Aug 14, 2004).

Wesemann, Daniel. “GCIA logs/Raw practicals and spp_stream4.” Jan 5, 2003. URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00018.html (Aug 10, 2004).

Zalewski, Michal [1]: “p0f file listing” Jul 15, 2004. URL http://www.stearns.org/p0f/ (Aug 27, 2004).

Zalewski, Michal [2]: “README” Component of p0f load. Jul 15, 2004. URL http://www.stearns.org/p0f/ (Aug 27, 2004).

4.2 Detect Totals

Priority	Detect	Count	Direction
1	CHAT MSN message	169	egress
1	WEB-IIS ISAPI .ida attempt	2	ingress
1	WEB-CGI formmail arbitrary command execution attempt [sid] [arachNIDS]	2	ingress
2	RPC portmap mountd request UDP [sid] [arachNIDS]	16	ingress
2	SCAN Squid Proxy attempt [sid]	16	ingress
2	SCAN Proxy Port 8080 attempt [sid]	16	ingress
2	SCAN SOCKS Proxy attempt [help.undernet.org] [sid]	15	ingress
2	BAD-TRAFFIC same SRC/DST [sid] [BUGTRAQ]	12	ingress
2	WEB-FRONTPAGE /_vti_bin/ access [sid]	3	ingress
2	(http_inspect) BARE BYTE UNICODE ENCODING [arachNIDS]	2	ingress
2	WEB-IIS ISAPI .ida access [sid] [arachNIDS]	2	ingress
2	ATTACK-RESPONSES 403 Forbidden [sid]	2	egress
2	WEB-CGI formmail access [sid] [arachNIDS]	2	ingress
2	WEB-FRONTPAGE shtml.exe access [sid] [BUGTRAQ]	1	ingress
2	WEB-IIS _vti_inf access [sid]	1	ingress
2	WEB-FRONTPAGE _vti_rpc access [sid] [BUGTRAQ]	1	ingress
3	BACKDOOR Q access [sid] [arachNIDS]	21	ingress
3	BAD-TRAFFIC tcp port 0 traffic [sid]	16	ingress
3	BAD-TRAFFIC ip reserved bit set [sid]	8	ingress
N/A	(http_inspect) NON-RFC HTTP DELIMITER	1	ingress

Table 12 Log File 2002.10.18 Detects

4.3 FrontPage Session Detects and Datagrams

The detects and datagrams associated with 2.3 Detect 1 - WEB-FRONTPAGE _vti_rpc access. Snort detects:

```
[**] [1:990:6] WEB-IIS _vti_inf access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
11/17/02-20:33:17.856507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x1AD
61.140.72.65:51597 -> 170.129.50.3:80 TCP TTL:45 TOS:0x0 ID:20626 IpLen:20 DgmLen:415
DF
***AP*** Seq: 0xB47BDF9B  Ack: 0xC12EBE19  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 84578131 134089

[**] [1:962:9] WEB-FRONTPAGE shtml.exe access [**]
```

```
[Classification: access to a potentially vulnerable web application] [Priority: 2]
11/17/02-20:33:20.336507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x201
61.140.72.65:51597 -> 170.129.50.3:80 TCP TTL:45 TOS:0x0 ID:20630 IpLen:20 DgmLen:499
DF
***AP*** Seq: 0xB47BE106  Ack: 0xC12ECE7B  Win: 0x3890  TcpLen: 32
TCP Options (3) => NOP NOP TS: 84578379 134194
[Xref => http://cgi.nessus.org/plugins/dump.php3?id=10405][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2000-0709][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2000-0413][Xref =>
http://www.securityfocus.com/bid/1608][Xref => http://www.securityfocus.com/bid/1174]

[**] [1:1288:6] WEB-FRONTPAGE /_vti_bin/ access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
11/17/02-20:33:20.336507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x201
61.140.72.65:51597 -> 170.129.50.3:80 TCP TTL:45 TOS:0x0 ID:20630 IpLen:20 DgmLen:499
DF
***AP*** Seq: 0xB47BE106  Ack: 0xC12ECE7B  Win: 0x3890  TcpLen: 32
TCP Options (3) => NOP NOP TS: 84578379 134194

[**] [1:937:7] WEB-FRONTPAGE _vti_rpc access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
11/17/02-20:33:20.336507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x201
61.140.72.65:51597 -> 170.129.50.3:80 TCP TTL:45 TOS:0x0 ID:20630 IpLen:20 DgmLen:499
DF
***AP*** Seq: 0xB47BE106  Ack: 0xC12ECE7B  Win: 0x3890  TcpLen: 32
TCP Options (3) => NOP NOP TS: 84578379 134194
[Xref => http://www.securityfocus.com/bid/2144]

[**] [119:13:1] (http_inspect) NON-RFC HTTP DELIMITER [**]
11/17/02-20:33:20.336507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x6B
61.140.72.65:51597 -> 170.129.50.3:80 TCP TTL:45 TOS:0x0 ID:20631 IpLen:20 DgmLen:93
DF
***AP*** Seq: 0xB47BE2C5  Ack: 0xC12ECE7B  Win: 0x3890  TcpLen: 32
TCP Options (3) => NOP NOP TS: 84578380 134194
```

Tcpdump decode:

```
20:33:17.856507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4 (0x0800), length
429: IP (tos 0x0, ttl  45, id 20626, offset 0, flags [DF], length: 415)
61.140.72.65.51597 > 170.129.50.3.80: P [bad tcp cksum f5ea (->192b)!]
3028017051:3028017414(363) ack 3241066009 win 5840 <nop,nop,timestamp 84578131 134089>
        0x0000:  4500 019f 5092 4000 2d06 9975 3d8c 4841  E...P.@.-..u=.HA
        0x0010:  aa81 3203 c98d 0050 b47b df9b c12e be19  ..2....P.{......
        0x0020:  8018 16d0 f5ea 0000 0101 080a 050a 8f53  ...............S
        0x0030:  0002 0bc9 4745 5420 2f5f 7674 695f 696e  ....GET./_vti_in
        0x0040:  662e 6874 6d6c 2048 5454 502f 312e 300d  f.html.HTTP/1.0.
        0x0050:  0a44 6174 653a 2057 6564 2c20 3136 204a  .Date:.Wed,.16.J
        0x0060:  616e 2032 3030 3220 3035 3a33 373a 3135  an.2002.05:37:15
        0x0070:  2047 4d54 0d0a 4d69 6d65 2d56 6572 7369  .GMT..Mime-Versi
        0x0080:  6f6e 3a20 312e 300d 0a41 6363 6570 743a  on:.1.0..Accept:
        0x0090:  202a 2f2a 0d0a 5573 6572 2d41 6765 6e74  .*/*..User-Agent
        0x00a0:  3a20 4d6f 7a69 6c6c 612f 322e 3020 2863  :.Mozilla/2.0.(c
        0x00b0:  6f6d 7061 7469 626c 653b 204d 5320 4672  ompatible;.MS.Fr
        0x00c0:  6f6e 7450 6167 6520 342e 3029 0d0a 4163  ontPage.4.0)..Ac
        0x00d0:  6365 7074 3a20 6175 7468 2f73 6963 696c  cept:.auth/sicil
        0x00e0:  790d 0a43 6f6e 7465 6e74 2d4c 656e 6774  y..Content-Lengt
        0x00f0:  683a 2030 0d0a 5072 6167 6d61 3a20 6e6f  h:.0..Pragma:.no
        0x0100:  2d63 6163 6865 0d0a 5669 613a 2031 2e30  -cache..Via:.1.0
        0x0110:  2070 726f 7879 333a 3830 3830 2028 5371  .proxy3:8080.(Sq
        0x0120:  7569 642f 322e 342e 5354 4142 4c45 3129  uid/2.4.STABLE1)
        0x0130:  0d0a 582d 466f 7277 6172 6465 642d 466f  ..X-Forwarded-Fo
        0x0140:  723a 2031 302e 3139 322e 3230 382e 3932  r:.10.192.208.92
        0x0150:  0d0a 486f 7374 3a20 7777 772e 5858 5858  ..Host:.www.XXXX
```

```
        0x0160:  5858 5858 0d0a 4361 6368 652d 436f 6e74  XXXX..Cache-Cont
        0x0170:  726f 6c3a 206d 6178 2d61 6765 3d32 3539  rol:.max-age=259
        0x0180:  3230 300d 0a43 6f6e 6e65 6374 696f 6e3a  200..Connection:
        0x0190:  206b 6565 702d 616c 6976 650d 0a0d 0a    .keep-alive....
20:33:20.336507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4 (0x0800), length
513: IP (tos 0x0, ttl  45, id 20630, offset 0, flags [DF], length: 499)
61.140.72.65.51597 > 170.129.50.3.80: P [bad tcp cksum 99c5 (->bd05)!]
3028017414:3028017861(447) ack 3241070203 win 14480 <nop,nop,timestamp 84578379
134194>
        0x0000:  4500 01f3 5096 4000 2d06 991d 3d8c 4841  E...P.@.-...=.HA
        0x0010:  aa81 3203 c98d 0050 b47b e106 c12e ce7b  ..2....P.{.....{
        0x0020:  8018 3890 99c5 0000 0101 080a 050a 904b  ..8............K
        0x0030:  0002 0c32 504f 5354 202f 5f76 7469 5f62  ...2POST./_vti_b
        0x0040:  696e 2f73 6874 6d6c 2e65 7865 2f5f 7674  in/shtml.exe/_vt
        0x0050:  695f 7270 6320 4854 5450 2f31 2e30 0d0a  i_rpc.HTTP/1.0..
        0x0060:  4461 7465 3a20 5765 642c 2031 3620 4a61  Date:.Wed,.16.Ja
        0x0070:  6e20 3230 3032 2030 353a 3337 3a31 3720  n.2002.05:37:17.
        0x0080:  474d 540d 0a4d 696d 652d 5665 7273 696f  GMT..Mime-Versio
        0x0090:  6e3a 2031 2e30 0d0a 5573 6572 2d41 6765  n:.1.0..User-Age
        0x00a0:  6e74 3a20 4d53 4672 6f6e 7450 6167 652f  nt:.MSFrontPage/
        0x00b0:  342e 300d 0a41 6363 6570 743a 2061 7574  4.0..Accept:.aut
        0x00c0:  682f 7369 6369 6c79 0d0a 436f 6e74 656e  h/sicily..Conten
        0x00d0:  742d 4c65 6e67 7468 3a20 3431 0d0a 436f  t-Length:.41..Co
        0x00e0:  6e74 656e 742d 5479 7065 3a20 6170 706c  ntent-Type:.appl
        0x00f0:  6963 6174 696f 6e2f 782d 7777 772d 666f  ication/x-www-fo
        0x0100:  726d 2d75 726c 656e 636f 6465 640d 0a58  rm-urlencoded..X
        0x0110:  2d56 6572 6d65 6572 2d43 6f6e 7465 6e74  -Vermeer-Content
        0x0120:  2d54 7970 653a 2061 7070 6c69 6361 7469  -Type:.applicati
        0x0130:  6f6e 2f78 2d77 7777 2d66 6f72 6d2d 7572  on/x-www-form-ur
        0x0140:  6c65 6e63 6f64 6564 0d0a 5072 6167 6d61  lencoded..Pragma
        0x0150:  3a20 6e6f 2d63 6163 6865 0d0a 5669 613a  :.no-cache..Via:
        0x0160:  2031 2e30 2070 726f 7879 333a 3830 3830  .1.0.proxy3:8080
        0x0170:  2028 5371 7569 642f 322e 342e 5354 4142  .(Squid/2.4.STAB
        0x0180:  4c45 3129 0d0a 582d 466f 7277 6172 6465  LE1)..X-Forwarde
        0x0190:  642d 466f 723a 2031 302e 3139 322e 3230  d-For:.10.192.20
        0x01a0:  382e 3932 0d0a 486f 7374 3a20 7777 772e  8.92..Host:.www.
        0x01b0:  5858 5858 5858 5858 0d0a 4361 6368 652d  XXXXXXXX..Cache-
        0x01c0:  436f 6e74 726f 6c3a 206d 6178 2d61 6765  Control:.max-age
        0x01d0:  3d32 3539 3230 300d 0a43 6f6e 6e65 6374  =259200..Connect
        0x01e0:  696f 6e3a 206b 6565 702d 616c 6976 650d  ion:.keep-alive.
        0x01f0:  0a0d 0a                                  ...
20:33:20.336507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4 (0x0800), length
107: IP (tos 0x0, ttl  45, id 20631, offset 0, flags [DF], length: 93)
61.140.72.65.51597 > 170.129.50.3.80: P [bad tcp cksum ddd7 (->847)!]
3028017861:3028017902(41) ack 3241070203 win 14480 <nop,nop,timestamp 84578380 134194>
0x0000:  4500 005d 5097 4000 2d06 9ab2 3d8c 4841  E..]P.@.-...=.HA
        0x0010:  aa81 3203 c98d 0050 b47b e2c5 c12e ce7b  ..2....P.{.....{
        0x0020:  8018 3890 ddd7 0000 0101 080a 050a 904c  ..8............L
        0x0030:  0002 0c32 6d65 7468 6f64 3d73 6572 7665  ...2method=serve
        0x0040:  722b 7665 7273 696f 6e3a 342e 302e 322e  r+version:4.0.2.
        0x0050:  3236 3131 0a25 3265 3236 3131 0a         2611.%2e2611.
```