Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment
Version 4.0

陳解濤 (CHEN JieTao)

2005/3/10

Prepared with OpenOffice.org V 1.1.3 on Novell SuSE Linux V 9.2

Table of Contents

1.Executive Summary

To assess the network security status of the University, I have analyzed a network log in detail. The log data were collected on 2002/6/13 by Snort, the de facto standard intrusion detection tool.

So far no solid signs of system compromise were found. However, it is far from certain to conclude that the University network is safe. There were a lot of attempted attacks and information gathering behaviors from the wild worth looking into. For example, there were BACKDOOR Q packets sent to 35 different University IP addresses, SHELLCODE x86 NOOP attacks, various kinds of attacks against IIS (Microsoft Internet Information Server), etc.. Information gathering attempts included DNS named version query targeting 40 different University IP addresses. Because the log itself was sanitized in certain ways, I am unable to decide that those attempts were successful or not. It is my suggestion that the University need to invest more resource to the network security team to further check the status.

Besides those attacks and likely malicious attempts from outside, cases of potential network policy violation from inside the University network were identified.

Specifically, there was a host, 46.5.180.250, which needs immediate checking. Significant amount of outbound traffic captured by IDS sensor was from this system, including P2P requests, MSN Messenger chat and many HTTP transactions. Some of the HTTP requests may have been sent by malware such as Gator running on the system.

It has been widely publicized that universities have been heaven of warez. It is true that academic institutions should value information sharing. But some people take that to a negative extreme. They abuse trust by the community and use the academic resource to “share” licensed/copyrighted information properties products such as music, movie and software. This has caused legal actions resulting in litigations against individuals and institutions. So the University should run a Security Awareness program and educate all of its users about the serious nature of it.

The Internet has seen more and more threats from the wild. Significant amount of malicious traffic have been witnesses from outside. Although no solid evidence of security compromises have been identified so far, without a solid security policy and serious implementation, the University network may not withstand a sudden strike. Nowadays, like almost any other organizations, the University heavily relies on the network infrastructure to behave. So we strongly recommend that a thorough review of network security/privacy policy of the University be started as soon as possible.

2.Detailed Analysis

2.1 Scenario Identification

This analysis is based on the log found at http://isc.sans.org/logs/Raw/2002.5.13. According to the assignment, scenario "selection should be based on the date in the file names". To find out if this is the case, I checked the earliest and latest time stamp in the log are as follows:

```
> tcpdump -nr 2002.5.13 -tttt | awk '{print $1, $2}' | head -1
reading from file 2002.5.13, link-type EN10MB (Ethernet)
2002-06-13 00:00:52.954488
> tcpdump -nr 2002.5.13 -tttt | awk '{print $1, $2}' | tail -1
reading from file 2002.5.13, link-type EN10MB (Ethernet)
2002-06-13 23:57:24.594488
```

So it is actually one-day data captured on 2002/6/13, not 2002/5/13.

Looking into the same directory http://isc.sans.org/logs/Raw/, one can find that there are other interesting filenames including 2002.4.31 and 2002.9.31. Unsurprising to anyone, there are no such dates as "2002/4/31" or "2002/9/31".

It appears to me that the script that was used to generate the log filename used 0 - 11 to denote the months. Thus, January is the 0th month, June is the 5th month and October is the 9th month. I randomly checked several other logs, and found no exceptions to my theory so far.

And one can easily find out the version of Snort used to generate the log:

```
> file 2002.5.13
2002.5.13: tcpdump capture file (little-endian) - version 2.4 (Ethernet, capture length 1514)
```

However, the Snort rules used were unknown. Also according to the README file located in the raw log file directory (http://isc.sans.org/logs/Raw/README), the logs were sanitized in various ways before they were uploaded for public review. Understandably, the IP addresses and server names inside University network were "munged"; and checksums of the packets were "modified to prevent clever people from discovering the original IP addresses"; captured packets for many other protocols such as ICMP, DNS, SMTP and HTTP were removed. As I will show, this incompleteness nature of the Snort log makes me difficult to draw a confirmed conclusion about the security status of the University network.

2.2 Network Topology

First of all, we need to get the unique MAC addresses in question, i.e., what network devices the IDS sensor was monitoring. I ran

```
> tcpdump -ner 2002.5.13 | awk '{print $2}' | sort -u
reading from file 2002.5.13, link-type EN10MB (Ethernet)
00:00:0c:04:b2:33
```

```
00:03:e3:d9:26:c0
```

So we have two MAC addresses here: 00:00:0c:04:b2:33 and 00:03:e3:d9:26:c0.

Notice that these are exactly the same as found in several others' assignments, such as Ian Marks (http://www.giac.org/certified_professionals/practicals/gcia/0760.php) and Joel Esler (http://www.giac.org/certified_professionals/practicals/gcia/0749.php). According to their findings, these MAC addresses belong to CISCO. For simplicity's sake, let's denote the two devices by the last hex of their MAC addresses, as CISCO-33 and CISCO-C0 respectively.

To find out if there are anomalous captured packets with the same source and destination, use command line

```
> tcpdump -ner 2002.5.13 | awk '{if ($2 == $4) print $2}'
reading from file 2002.5.13, link-type EN10MB (Ethernet)
```

and get no results. So every packet in the captured log was either from CISCO-33 to CISCO-C0 or from CISCO-C0 to CISCO-33, and no packets were abnormal in this level.

Then we looked at traffic flow through these two devices. To get there we need to look at the source and destination IP addresses associated with each of them.

As said, there are only two possible flow for each packet, from CISCO-33 to CISCO-C0 or reverse. So the following 4 command line can exhaust all possibilities on source and destination IP addresses connected to each device:

```
> tcpdump -ner 2002.5.13 ether src 0:0:c:4:b2:33 | awk '{print $11}' | awk -F .
'{print $1 "." $2 "." $3 "." $4}' | sort | uniq -c | sort -r
reading from file 2002.5.13, link-type EN10MB (Ethernet)
   4353 46.5.180.250
      5 46.5.180.133
```

These two are source IP addresses for the packets sent from CISCO-33.

```
> tcpdump -ner 2002.5.13 ether src 0:0:c:4:b2:33 | awk '{print $13}' | awk -F .
'{print $1 "." $2 "." $3 "." $4}' | sort | uniq -c  | sort -r
reading from file 2002.5.13, link-type EN10MB (Ethernet)
   2684 64.154.80.51
   1074 204.178.98.77
     90 207.68.162.250
        ...
      1 149.156.118.119
      1 141.35.14.47
```

These are destination IP addresses for the packets sent from CISCO-33.

```
> tcpdump -ner 2002.5.13 ether dst 0:0:c:4:b2:33 | awk '{print $11}' | awk -F .
'{print $1 "." $2 "." $3 "." $4}' | sort | uniq -c | sort -r
reading from file 2002.5.13, link-type EN10MB (Ethernet)
     35 255.255.255.255
```

```
     32 207.150.192.12
     21 207.188.7.150
        ...
      1 168.234.191.29
      1 12.40.107.250
```

These are source IP addresses for the packets sent from CISCO-C0.

```
> tcpdump -ner 2002.5.13 ether dst 0:0:c:4:b2:33 | awk '{print $13}' | awk -F .
'{print $1 "." $2 "." $3 "." $4}' | sort | uniq -c | sort -r

reading from file 2002.5.13, link-type EN10MB (Ethernet)

    194 46.5.180.250
     46 46.5.180.133
     16 46.5.238.166
        ...
      1 46.5.101.185
      1 46.5.0.78
```

These are destination IP addresses for the packets sent from CISCO-C0.

We can see that traffic from CISCO-33 was sent from two distinct IP addresses: 46.5.180.250 and 46.5.180.133, and toward a wide range of networks; traffic from CISCO-C0 was from a wide range of networks to network 46.5.0.0/32. With this information, we can safely draw conclusion that 46.5.0.0/32 belongs to the University network; CISCO-33 was located insider of the University network, while CISCO-C0 was closer to the network border, and the Snort sensor was located between these two devices, probably via a hub or spanning port.

We saw that packets from CISCO-33 have only two source IP addresses. Regarding 46.5.180.133, the lesser talker, let's find out source ports of those packets sent from it and destination posts of those sent to it,

```
> tcpdump -ner 2002.5.13 ip src 46.5.180.133 | awk '{print $11}' | awk -F \.
'{print $5}' | sort | uniq -c

reading from file 2002.5.13, link-type EN10MB (Ethernet)

      5 80
```

So source port of the packets sent from 46.5.180.133 is 80, which is standard HTTP server port. But port 80 does not guarantee a packet to be HTTP traffic. We need to get down to the payload of the packet to be able to tell.

```
> tcpdump -ner 2002.5.13 ip dst 46.5.180.133 | awk '{print $13}' | awk -F \.
'{print $5}' | sort | uniq -c

reading from file 2002.5.13, link-type EN10MB (Ethernet)

     10 21:
     36 80:
```

Two distinct destination ports exist for the packets sent to 46.5.180.133, 21 and 80. Port 21 is standard FTP server port. But again, without looking inside the packet, we can not tell for sure that these packets were HTTP or FTP traffic or not.

Further checking on the payload of the seemingly HTTP packets sent from this host showed that it is indeed a HTTP server. For instance, the first one is as follows:

```
> tcpdump -Xner 2002.5.13 ip src 46.5.180.133 -c 1
reading from file 2002.5.13, link-type EN10MB (Ethernet)
20:06:02.954488 00:00:0c:04:b2:33 > 00:03:e3:d9:26:c0, ethertype IPv4 (0x0800),
length 588: IP 46.5.180.133.80 > 212.62.33.4.1978: P 3041588787:3041589309(522)
ack 2918340145 win 31856 <nop,nop,timestamp 7555720 84018737>
        0x0000:  4500 023e 3dfe 4000 3f06 29f5 2e05 b485  E..>=.@.?.).....
        0x0010:  d43e 2104 0050 07ba b54a f633 adf2 5631  .>!..P...J.3..V1
        0x0020:  8018 7c70 0b88 0000 0101 080a 0073 4a88  ..|p.........sJ.
        0x0030:  0502 0631 4854 5450 2f31 2e31 2034 3033  ...1HTTP/1.1.403
        0x0040:  2046 6f72 6269 6464 656e 0d0a 4461 7465  .Forbidden..Date
        0x0050:  3a20 5468 752c 2031 3320 4a75 6e20 3230  :.Thu,.13.Jun.20
        0x0060:  3032 2030 313a 3030 3a32 3320 474d 540d  02.01:00:23.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 206d  Red.Hat/Linux).m
        0x00a0:  6f64 5f6a 6b20 6d6f 645f 7373 6c2f 322e  od_jk.mod_ssl/2.
        0x00b0:  362e 3620 4f70 656e 5353 4c2f 302e 392e  6.6.OpenSSL/0.9.
        0x00c0:  3561 2050 4850 2f34 2e30 2e31 706c 3220  5a.PHP/4.0.1pl2.
        0x00d0:  6d6f 645f 7065 726c 2f31 2e32 3420 4672  mod_perl/1.24.Fr
        0x00e0:  6f6e 7450 6167 652f 342e 302e 342e 330d  ontPage/4.0.4.3.
        0x00f0:  0a43 6f6e 6e65 6374 696f 6e3a 2063 6c6f  .Connection:.clo
        0x0100:  7365 0d0a 436f 6e74 656e 742d 5479 7065  se..Content-Type
        0x0110:  3a20 7465 7874 2f68 746d 6c3b 2063 6861  :.text/html;.cha
        0x0120:  7273 6574 3d69 736f 2d38 3835 392d 310d  rset=iso-8859-1.
        0x0130:  0a0d 0a3c 2144 4f43 5459 5045 2048 544d  ...<!DOCTYPE.HTM
        0x0140:  4c20 5055 424c 4943 2022 2d2f 2f49 4554  L.PUBLIC."-//IET
        0x0150:  462f 2f44 5444 2048 544d 4c20 322e 302f  F//DTD.HTML.2.0/
        0x0160:  2f45 4e22 3e0a 3c48 544d 4c3e 3c48 4541  /EN">.<HTML><HEA
        0x0170:  443e 0a3c 5449 544c 453e 3430 3320 466f  D>.<TITLE>403.Fo
        0x0180:  7262 6964 6465 6e3c 2f54 4954 4c45 3e0a  rbidden</TITLE>.
        0x0190:  3c2f 4845 4144 3e3c 424f 4459 3e0a 3c48  </HEAD><BODY>.<H
        0x01a0:  313e 466f 7262 6964 6465 6e3c 2f48 313e  1>Forbidden</H1>
        0x01b0:  0a59 6f75 2064 6f6e 2774 2068 6176 6520  .You.don't.have.
        0x01c0:  7065 726d 6973 7369 6f6e 2074 6f20 6163  permission.to.ac
        0x01d0:  6365 7373 202f 0a6f 6e20 7468 6973 2073  cess./.on.this.s
        0x01e0:  6572 7665 722e 3c50 3e0a 3c48 523e 0a3c  erver.<P>.<HR>.<
        0x01f0:  4144 4452 4553 533e 4170 6163 6865 2f31  ADDRESS>Apache/1
```

```
0x0200:  2e33 2e31 3220 5365 7276 6572 2061 7420  .3.12.Server.at.
0x0210:  7777 772e 5858 5858 2e63 6f6d 2050 6f72  www.XXXX.com.Por
0x0220:  7420 3830 3c2f 4144 4452 4553 533e 0a3c  t.80</ADDRESS>.<
0x0230:  2f42 4f44 593e 3c2f 4854 4d4c 3e0a       /BODY></HTML>.
```

The data indicates that the host ran Apache 1.3.12 on RedHat Linux system. And the packet contains a HTTP 403 access denied error.

Notice that besides its legitimate use, HTTP 403 error page can also be used by malicious party to gather information of a web server. When they find out the server type and version, mostly with operating system type and version as well, they can exploit security vulnerabilities found on the specific version of the HTTP server and specific version of the operating system.

With only incoming FTP anonymous login attempts and no other evidence, we can not be sure whether it actually hosts a FTP server, or simply some attackers tried to probe and/or attack it.

We can then find out who this host talked with,

```
> tcpdump -ner 2002.5.13 ip src 46.5.180.133 | awk '{print $13}' | awk -F \.
'{print $1 "." $2 "." $3 "." $4}' | sort | uniq

reading from file 2002.5.13, link-type EN10MB (Ethernet)

195.29.138.237

212.62.33.4
```

46.5.180.133 sent packets to these 2 IP addresses from its port 80.

```
> tcpdump -ner 2002.5.13 'ip dst 46.5.180.133 and tcp port 80' | awk '{print
$11}' | awk -F \. '{print $1 "." $2 "." $3 "." $4}' | sort | uniq

reading from file 2002.5.13, link-type EN10MB (Ethernet)

131.188.134.2

165.21.47.198

207.152.116.120

208.10.255.66

217.235.152.207

218.145.63.95

62.251.89.129

66.124.11.133

67.82.30.123
```

46.5.180.133 received packets from these IP addresses to its port 80.

```
> tcpdump -ner 2002.5.13 'ip dst 46.5.180.133 and tcp port 21' | awk '{print
$11}' | awk -F \. '{print $1 "." $2 "." $3 "." $4}' | sort | uniq

reading from file 2002.5.13, link-type EN10MB (Ethernet)

155.245.44.32

202.37.96.11

208.138.3.3

210.175.81.29
```

211.23.87.222

212.150.108.15

213.76.118.93

These are the IP addresses that tried to do anonymously FTP login to 46.5.180.133.

Thus we can get the link graph of 46.5.180.133 as follows:

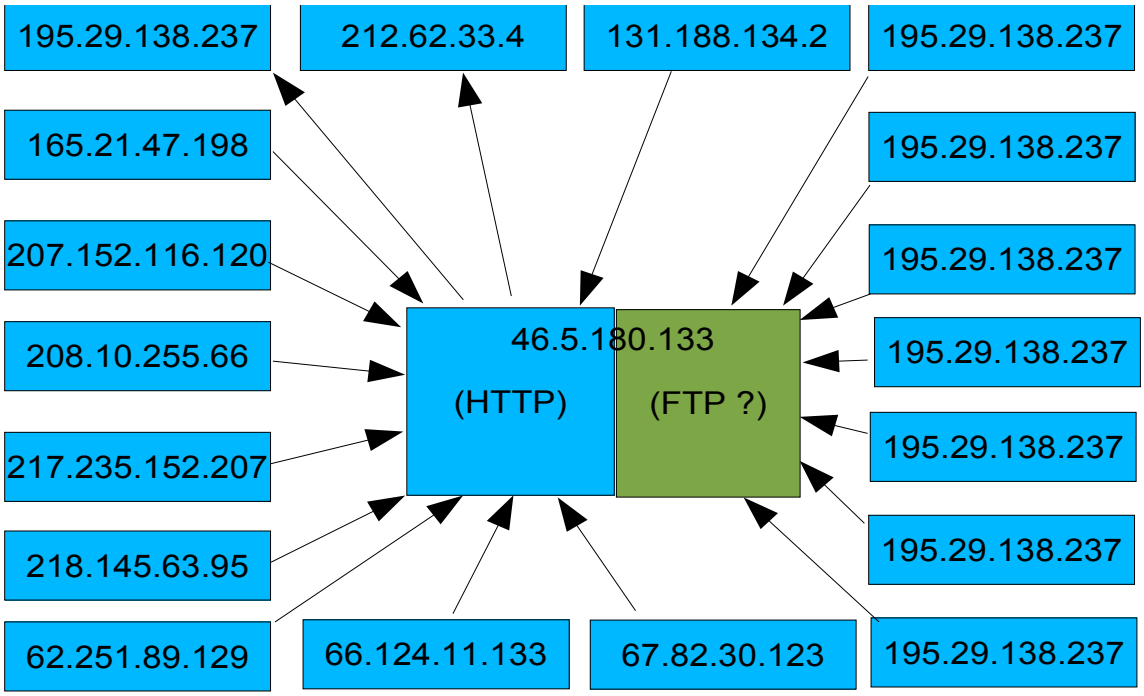

Figure 1 Link Graph for 46.2.180.133

The other University IP address, 46.5.180.250, will be checked in part 3 of "Detects in Detail" below.

With all the above information, we can now draft the following network typology graph:

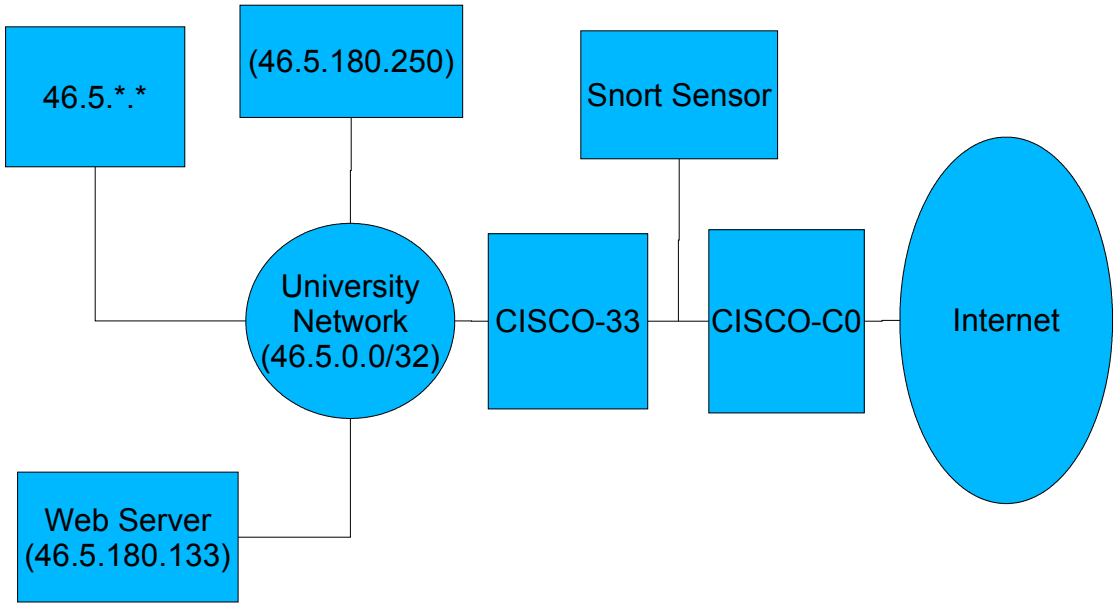

Figure 2 Network Topology

2.3 Overview of Alerts

Running Snort version 2.2.0 with the default rules (all enabled), with the help of SnortSnarf, we can get summary report of the alerts.

```
> snort -r 2002.5.13 -k none -A full -dyev -c /etc/snort/snort.conf -h 46.5.0.0/32

...

...

...
```

As mentioned, checksums of the packets were scrambled by SANS/GIAC, so option "-k none" was used to prevent Snort from verifying checksums. Otherwise, Snort will not produce any alerts.

```
> ./snortsnarf.pl -rs -d /tmp/output/ /var/log/snort/alert
```

SnortSnarf generated HTML format of alert reports to directory /tmp/output.

One should note that this method by no means is perfect or precise without further checking into the log. For example, the signature "P2P Outbound GNUTella client request" is actually duplicate of signature "P2P GNUTella client request"; "WEB-FRONTPAGE _vti_rpc access" is also duplicate of "WEB-FRONTPAGE shtml.exe access" here, and so forth.

Nonetheless, the report still gives us a pretty good picture after all. Table 1 below shows the overall statistics we got from HTML summary report generated by Snort and SnortSnarf.

Priority	Signature	# Alerts	# Sources	# Dests
1	P2P GNUTella client request [sid]	133	1	69
1	P2P Outbound GNUTella client request [sid]	133	1	69
1	SHELLCODE x86 NOOP [sid]	9	3	1
1	WEB-IIS cmd.exe access [sid]	6	1	5
1	CHAT MSN message [sid]	2	1	1
1	WEB-IIS ISAPI .ida attempt [sid] [arachNIDS]	1	1	1
2	DNS named version attempt [sid] [arachNIDS]	40	8	40
2	WEB-IIS view source via translate header [sid] [arachNIDS]	19	2	1
2	(http_inspect) NON-RFC HTTP DELIMITER [BUGTRAQ]	12	4	6
2	WEB-CGI redirect access [sid] [BUGTRAQ]	8	1	2
2	WEB-FRONTPAGE /_vti_bin/ access [cgi.nessus.org] [sid]	8	6	1
2	WEB-MISC http directory traversal [sid] [arachNIDS]	6	1	5
2	WEB-FRONTPAGE _vti_inf.html access [cgi.nessus.org] [sid]	5	4	1
2	(http_inspect) WEBROOT DIRECTORY TRAVERSAL [arachNIDS]	5	1	5
2	WEB-FRONTPAGE shtml.exe access [sid] [BUGTRAQ]	5	4	1
2	ATTACK-RESPONSES 403 Forbidden [sid]	5	1	2
2	WEB-FRONTPAGE _vti_rpc access [sid] [BUGTRAQ]	5	4	1
2	WEB-IIS %2E-asp access [sid] [BUGTRAQ]	4	1	1
2	DNS zone transfer TCP [sid] [arachNIDS]	2	2	1
2	WEB-MISC Invalid HTTP Version String [sid] [BUGTRAQ]	1	1	1
2	WEB-IIS ISAPI .ida access [sid] [arachNIDS]	1	1	1
2	WEB-CGI formmail access [sid] [arachNIDS]	1	1	1
3	BAD-TRAFFIC tcp port 0 traffic [sid]	48	3	3
3	BACKDOOR Q access [sid] [arachNIDS]	35	1	35
3	POLICY FTP anonymous login attempt [sid]	10	7	1
3	BAD-TRAFFIC ip reserved bit set [sid]	2	1	2
N/A	(http_inspect) BARE BYTE UNICODE ENCODING	850	1	9
N/A	(http_inspect) APACHE WHITESPACE (TAB)	22	1	3
N/A	(http_inspect) OVERSIZE REQUEST-URI DIRECTORY	18	1	2
N/A	(snort_decoder) WARNING: TCP Data Offset is less than 5!	10	2	2
N/A	(http_inspect) DOUBLE DECODING ATTACK	7	1	4
N/A	(http_inspect) IIS UNICODE CODEPOINT ENCODING	2	1	1

Table 1 Alert Summary Generated by Snort and SnortSnarf

2.4 Detects in Detail

Here I present detailed analysis of 3 detects that I believe deserve more scrutiny. Detect 1 is about BACKDOOR Q tack. Detect 2 is about WEB-IIS cmd.exe access. Part 3 talks about a combination of MSNMS, GNUTELLA and problematic HTTP traffic sent from inside University network. Part 3 actually focuses on the most noisy host from inside. Although these kinds of traffic were not direct attacks from outside, they are interesting both from a technical and ethical points of view. I will get down to the point in more detail in the section.

2.4.1 Detect 1: BACKDOOR Q access

There were 35 BACKDOOR Q access alerts captured. 35 IP addresses were found targeted exactly once per each. For example,

```
[**] [1:184:6] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
06/13/02-12:50:32.114488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
255.255.255.255:31337 -> 46.5.89.150:515 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]
```

We can get a list of targeted IP addresses using following command line:

```
tcpdump -ner 2002.5.13 ip src 255.255.255.255 | awk '{print $13}'|sort
```

And the list of target IP addresses is

46.5.0.78.515	46.5.104.223.515	46.5.149.178.515
46.5.15.4.515	46.5.150.43.515	46.5.153.152.515
46.5.153.9.515	46.5.157.64.515	46.5.157.70.515
46.5.170.206.515	46.5.174.176.515	46.5.184.183.515
46.5.189.222.515	46.5.190.242.515	46.5.198.168.515
46.5.198.25.515	46.5.205.50.515	46.5.226.167.515
46.5.227.99.515	46.5.244.41.515	46.5.247.13.515
46.5.253.225.515	46.5.253.3.515	46.5.28.11.515
46.5.30.224.515	46.5.42.32.515	46.5.46.82.515
46.5.48.70.515	46.5.73.211.515	46.5.74.133.515
46.5.76.184.515	46.5.76.25.515	46.5.87.106.515
46.5.87.80.515	46.5.89.150.515	

Table 2 IP Addresses targeted by BACKDOOR Q

As we can see, destination port was only 515. And destination IP addresses covered almost all possible subnet, from 46.5.0.* to 45.5.253.*. The attacks may be real or just wildly scan for infected hosts; it is also possible that the some of the packets were used to cover up others. We just can not be sure.

Attack Description

According to http://www.whitehats.com/info/IDS203,

```
Q is a remote access and redirection trojan that employs strong encryption. It
allows for the execution of remote commands as root by sending a raw
tcp/icmp/udp packet. This signature watches for the source address
255.255.255.255, which should not appear in normal traffic. The content of the
packet is the command to run as root - and is arbitrary.

Author's webpage is currently http://mixter.warrior2k.com/.
```

BACKDOOR Q is a sophisticated remote administration tool. It comes with a client/server pair, with server (daemon) installed on the infected machine, waiting for client (messenger) to deliver commands. The commands were sent one-way only, (generally) with class C broadcast address 255.255.255.255 as source IP address. According to Les Gordon (http://www.sans.org/resources/idfaq/qtrojan.php?printer=Y), this Trojan can compile and run on both UNIX and Windows systems.

Selection Criteria

- Very malicious, harmful level can be high. For instance, many BACKDOOR Q infected hosts may participate in a large scale of DDoS attack.
- The Q client hides itself by faking source address to be class C broadcast address. So it is difficult to trace the the origin of the attack, if firewall/router were not configured to prevent this kind of packets coming through.
- Without looking into the targeted hosts in detail, it is almost impossible to determine whether the targeted host is actually infected or not.
- The default Snort configuration does not include rules to detect it, so it may slip away from humans' eyes.

Detection Generation

```
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;)
```

This Snort rule is located in backdoor.rules file. Basically, it says that any ACK packets sent from network 255.255.255.0/24 to internal hosts with some payload will trigger the alert. Regarding the log we are investigating, all packets were sent from broadcast address 255.255.255.255.

Probability that Source Address was Spoofed

Of course the source address of BACKDOOR Q was spoofed, since it is a broadcast address here. As mentioned, the attack message delivery is one-way only, which means that the attacker does not need a direct response.

Correlation

This kind of attacks have been found long time ago.

- CAN-1999-0660 (under review)

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660

- IDS203 "TROJAN-ACTIVE-Q-TCP"

  http://www.whitehats.com/IDS/203

- What is the Q Trojan? by Les Gordon

  http://www.sans.org/resources/idfaq/qtrojan.php?printer=Y

Attack Mechanism

The BACKDOOR Q server was installed on the target host by any means, and listens on specific ports, such as 515 (printer spooler port assigned by IANA) in our case. On UNIX system, a program needs root privileges to be allowed to bind with ports number under 1024. So the corresponding server of Q clients in our case has root access, if there are any.

The attacker sends out crafted message to infected host to command the Q server. That is, the IP source address of the packet is 255.255.255.255, TCP source port is 31337, TCP target port is 515. The payload of the packets contains commands for the contacted Q server.

As port number 31337 is well known to be “signature” port a handful of Trojans, with BackOrifice to be probably the most famous one. So it is easy to detect the cases found in our log.

However, there are no reasons that the source port has to be 31337. The source IP does not have to be 255.255.255.255 either. It is possible that Q server judges whether a command message comes from Q client by using part of TCP payload. Knowing that those attackers are generally outsmart most of others, theoretically, it is very difficult, if not impossible, to detect all BACKDOOR Q.

Another possibility is that the Q client does not have to target specifically to an infected host. It is possible that Q server runs in promiscuous mode, which means that it can listens any packets passing through the local network. Thus a BACKDOOR Q command message targeted to any host on the same network can be picked up by the said Q server. This possibility makes detecting potential BACKDOOR Q infected hosts much more cumbersome.

In the sanitized log, no other suspicious traffic was found to be related with BACKDOOR Q attack. Unfortunately, for the reasons I discussed above, we still can not be sure whether the attacks were successful or not, nor can we decide whether there were BACKDOOR Q infected hosts inside University network or not. For example, what if the Q massager asked Q server to send some information somewhere through SMTP service? As all SMTP packets were removed from the log, we have no way to check that possibility.

Evidence of Active Targeting

As for the analysis above, basically, I can not tell whether the attacks were actively targeting the University network or not.

Severity

(3 + 5) – (2 + 3) = 3

Criticality	3	Unknown, all depends on whether there were infected hosts
Lethality	5	If infected, root access for the attacker
System Countermeasures	2	Unknown system level measures; if infected, difficult to detect and remove
Network Countermeasures	3	Poor filtering on University network border, but IDS detected the attack

Table 3 Severity Assessment for BACKDOOR Q Attack

Defensive Recommendations

As I argued, it is almost impossible to fully detect BACKDOOR Q Trojans. All we can do is try our best. always stick to best practice on University network setup and configurations.

2.4.2 Detect 2: WEB-IIS cmd.exe access

There were 6 WEB-IIS cmd.exe access alerts captured. 5 distinct IP addresses (46.5.180.151, 46.5.180.158, 46.5.180.145, 46.5.180.134, 46.5.180.150) were found targeted from one single IP address 64.51.28.93. For example,

```
[**] [1:1002:6] WEB-IIS cmd.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
06/13/02-07:47:20.834488 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x71
64.51.28.93:4430 -> 46.5.180.151:80 TCP TTL:114 TOS:0x0 ID:4770 IpLen:20
DgmLen:99 DF
***AP*** Seq: 0x75996DD9 Ack: 0xD3B171F Win: 0x2238 TcpLen: 20
```

The source IP appears to be registered to an ISP located in New Haven, CT. Samspade (http://www.samspade.org) returns:

```
64.51.28.93 = [ 64-51-28-93.client.dsl.net ]
OrgName:    DSL.net  Inc.
OrgID:      FTCI
Address:    545 Long Wharf Dr. 5th floow
City:       New Haven
StateProv:  CT
PostalCode: 06511
Country:    US
NetRange:   64.51.0.0 - 64.51.255.255
CIDR:       64.51.0.0/16
......
```

Attack Description

According to CVE-2000-0884,

```
IIS 4.0 and 5.0 allows remote attackers to read documents outside of the web
root, and possibly execute arbitrary commands, via malformed URLs that contain
UNICODE encoded characters, aka the "Web Server Folder Traversal" vulnerability.
```

Generally, web access is well limited to specific folders under web root on a web server. However, a so-called “Web Server Folder Traversal” vulnerability allows smart crafted URLS to be able to violate the rule. Checking the payload of the alerts showed that the attacks tried to launch cmd.exe, the Windows command line shell, to list contents of the current directory. This most likely were attempts to check if vulnerability exists on target hosts or not.

Selection Criteria

- Microsoft Windows holds big market share and Microsoft Windows systems are well known to have serious security problems. So they are made good targets from attackers.
- On many systems, necessary or not, IIS is installed and runs by default, while they are not taken good care of, especially in academic environments.
- Although many web servers are not mission-critical, they are the front-end to the outside world. Thus, if defaced, people may be less confident about the University in a whole.

Detection Generation

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS cmd.exe
access"; flow:to_server,established; content:"cmd.exe"; nocase; classtype:web-
application-attack; sid:1002; rev:6;)
```

This Snort rule is located in web-iis.rules file. This rule mandates that any HTTP message sent to web server containing substring “cmd.exe” (case-insensitive) will trigger the alert. Regarding the log we are investigating, all 6 packets were HTTP GET with URL to be “/scripts/..%5c%5c../winnt/system32/cmd.exe?/c+dir”.

Probability that Source Address was Spoofed

Generally speaking, An HTTP GET happens after 3-way TCP handshake. Unless the attacker can launch man-in-the-middle attack beforehand, it is very unlikely that the attacker with spoofed source address can achieve this. On the other hand, since the attacker wants to get the response, it is not likely that the source address of was spoofed.

It is possible that some real individual tried to do something to us, or the host of the source IP itself was infected by some Windows virus. Either of the cases, the University may need to contact the ISP about the case immediately.

Correlation

This kind of attacks are describe by Microsoft and other security organizations:

- CVE-2000-0884

http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0884

- Microsoft Security Bulletin (MS00-078)

  http://www.microsoft.com/technet/security/bulletin/MS00-078.mspx

Attack Mechanism

Again, from Microsoft Security Bulletin (MS00-078):

> Due to a canonicalization error in IIS 4.0 and 5.0, a particular type of malformed URL could be used to access files and folders that lie anywhere on the logical drive that contains the web folders. This would potentially enable a malicious user who visited the web site to gain additional privileges on the machine - specifically, it could be used to gain privileges commensurate with those of a locally logged-on user. Gaining these permissions would enable the malicious user to add, change or delete data, run code already on the server, or upload new code to the server and run it.
>
> The request would be processed under the security context of the IUSR_machinename account, which is the anonymous user account for IIS. Within the web folders, this account has only privileges that are appropriate for untrusted users. However, it is a member of the Everyone and Users groups and, as a result, the ability of the malicious user to access files outside the web folders becomes particularly significant. By default, these groups have execute permissions to most operating system commands, and this would give the malicious user the ability to cause widespread damage. Customers who have proactively removed the Everyone and Users groups from permissions on the server, or who are hosting the web folders on a different drive from the operating system, would be at significantly less risk from the vulnerability.

Normally, web access is limited to specific folders only. The web servers will check each incoming URL to make sure that the it does not violate this rule. If it is, a 403 access denied (forbidden) error or 404 Not Found error will be returned.

However, HTTP URL also allows “escaped” codes, using “%” and hexadecimal of a character to denote the character, such as “%20” denoting whitespace, “% 30” denoting character “0”. “%5c” is equivalent to backslash “\”. So the URL in question became “/scripts/..\\../winnt/system32/cmd.exe?/c+dir”, which is equivalent to “/scripts/../../winnt/system32/cmd.exe?/c+dir”, which gives the right path to cmd.exe. Unfortunately, earlier versions of IIS were programed to check URL integrity before the URL was decoded, so it missed the bad URL.

This exploit can be launched manually, or by viruses. Virus Nimda is one that can automatically send out this kind of malformed URLs. So it is also possible that 64.51.28.93, the host that attack came from was infected by virus such as Nimda.

Evidence of Active Targeting

The attack came from one IP address, targeting 6 distinct University owned IP, once for each. The targets were 46.5.180.134 – 46.5.180.158, appears to be in the same subnet. So the attacker seemed to be rather focused. So the possibility of active targeting is relatively high.

Severity

(3 + 4) – (2 + 3) = 2

Criticality	3	Not all traffic was logged. Hard to determine whether attacks were successful
Lethality	4	If infected, viruses may propagate inside the University
System Countermeasures	2	Unknown system patch level measures
Network Countermeasures	3	Poor filtering on University network border, but IDS detected the attack

Table 4 Severity Assessment for WEB-IIS cmd.exe Attack

Defensive Recommendations

Like other Microsoft products, IIS has pretty bad record on security. Try to avoid using IIS for web server. If for whatever reasons, IIS has to be used, IT department of the University should keep a keen eye on Microsoft Security Bulletin and apply security patches whenever they are released. However, there are times that security vulnerabilities are announced but patches may not be available yet. Then the best way to avoid successful attacks would probably be shutting down IIS or disconnecting the machine from the network.

2.4.3 Detect 3: GNUTELLA, MSNMS and Web Access

The target for this detect is not a specific attack, but for a specific host (46.5.180.250). As mentioned above, among all the 4811 captured alerts, 4353 (91.5%!)were sent from this IP address. Among those, 2 were MSNMS messages, 133 were GNUTELLA messages, and the rest 4128 were HTTP packets. I also stressed that those HTTP alerts are most likely false positives.

Selection Criteria

- This is definitely the TOP-TALKER, worth close checking. If we can identify false positives and adjust Snort detection rules accordingly, the IDS sensor and security analysts can be more focused on other malicious traffic.
- Many versions of MSN Messenger have been known to be exploitable.
- P2P programs such as GNUTELLA may violate University network use policy. If not dealing with properly, may result in legal actions targeted to the University by some third-parties.
- Abnormally high volume of advertisement behavior suggests that the host may be infected with malware (some prefer the name adware in this case).

Description, Detection and Correlations

(1) CHAT MSN message

MSN Messenger is a popular P2P style communication suite. Peers can exchange all kinds of information by using it. Of course, some viruses can spread through it as well. Using

```
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;)
```

from chat.rules, 2 alerts were recorded, such as:

```
[**] [1:540:11] CHAT MSN message [**]

[Classification: Potential Corporate Privacy Violation] [Priority: 1]

06/13/02-09:06:20.434488 0:0:C:4:B2:33 -> 0:3:E3:D9:26:C0 type:0x800 len:0xC8

46.5.180.250:64937 -> 64.4.12.170:1863 TCP TTL:124 TOS:0x0 ID:22581 IpLen:20
DgmLen:186 DF

***AP*** Seq: 0xFBAE913 Ack: 0x7B2C3CD6 Win: 0x21D8 TcpLen: 20
```

As a Microsoft product, it has a history of security vulnerabilities, such as

```
MS05-009

Vulnerability in PNG Processing Could Lead to Buffer Overrun (890261)

      http://www.microsoft.com/technet/security/Bulletin/MS05-009.mspx

MS04-010

Vulnerability in MSN Messenger Could Allow Information Disclosure (838512)

      http://www.microsoft.com/technet/security/Bulletin/MS04-010.mspx

MS02-022

Unchecked Buffer in MSN Chat Control Can Lead to Code Execution (Q321661)

      http://www.microsoft.com/technet/security/Bulletin/MS02-022.mspx
```

(2) P2P GNUTella client request

GNUTella was a popular P2P file exchange protocol. The agent string told us that the actual client program was Gnucleus version 1.6.0.0, one of several GNUTella front-end programs. The rule file is p2p.rules, specifically,

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"P2P GNUTella client
request"; flow:to_server,established; content:"GNUTELLA"; depth:8;
classtype:policy-violation; sid:1432; rev:6;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"P2P Outbound GNUTella client
request"; flow:to_server,established; content:"GNUTELLA CONNECT"; depth:40;
classtype:policy-violation; sid:556; rev:5;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"P2P GNUTella client
request"; flow:to_server,established; content:"GNUTELLA OK"; depth:40;
classtype:policy-violation; sid:557; rev:6;)
```

Totally 133 alerts were generated, for example,

```
[**] [1:1432:6] P2P GNUTella client request [**]

[Classification: Potential Corporate Privacy Violation] [Priority: 1]

06/13/02-05:57:59.214488 0:0:C:4:B2:33 -> 0:3:E3:D9:26:C0 type:0x800 len:0x6C

46.5.180.250:64397 -> 172.179.189.6:6388 TCP TTL:124 TOS:0x0 ID:22962 IpLen:20
DgmLen:94 DF

***AP*** Seq: 0x11FA7077 Ack: 0xAAD830 Win: 0x4320 TcpLen: 20
```

P2P file exchange programs can be of legitimate use, such as helping community for software release. And traditionally, universities are places that value information access and sharing. However, recent development has seen

possible legal risks on allowing P2P traffic without sound University network usage policy. The entertainment industry and some of the software companies may take legal actions against individuals or even institutions. For example, from InfoWorld news:

> Association has sued more than 8,400 people since September 2003
>
> **By Grant Gross, IDG News Service**
> January 27, 2005
>
> WASHINGTON - The Recording Industry Association of America (RIAA) has filed 717 new lawsuits against peer to peer (P-to-P) users allegedly trading music for free, the trade group announced Thursday.
>
> .....

The whole story can be read at the follwing URL,

> http://www.infoworld.com/article/05/01/27/HNriaanewsuits_1.html

The University may want to take the opportunity to deal with the issue.

(3) HTTP Traffic sent from 46.5.180.250

This class of traffic totaled around 86% of all alerts. As we looked at those alerts further, no solid evidence of malicious behavior was found.

First of all, I would like to see how many different IP addresses that these packets were sent to.

```
> tcpdump -ner 2002.5.13 ip src 46.5.180.250 | awk '{print $13}' | awk -F .
'{print $1 "." $2 "." $3 "." $4}' | sort | uniq | wc -l

reading from file 2002.5.13, link-type EN10MB (Ethernet)

120
```

So there were 120 distinct destinations for these captured packets. Then let's find out the top talkers of those 4128 HTTP alerts from the host

```
> tcpdump -ner 2002.5.13 ip src 46.5.180.250 | awk '{print $13}' | awk -F .
'{print $1 "." $2 "." $3 "." $4}' | sort | uniq -c | sort -r | awk '{if ($1 >
20) print $0}'

reading from file 2002.5.13, link-type EN10MB (Ethernet)

   2684 64.154.80.51

   1074 204.178.98.77

     90 207.68.162.250

     52 194.67.35.196

     48 194.67.23.251

     34 207.68.176.190

     21 208.254.63.69
```

As we see here, 2684 were sent to a single host 64.154.80.51, 1074 were sent to another one 204.178.98.77. To find out what these hosts are, samspade (http://www.samspade.org/t/whois) was used. See the following table for results:

IP	Frequency	Domain Name	Comments
64.154.80.51	2684	ehg.hitbox.com	Marketing tool Hitbox Gateway
204.178.98.77	1074	ww1.joboptions.com	Job search service
207.68.162.250	90	www.sea1.hotmail.com	Popular HOTMAIL service
194.67.35.196	52	forwarder6.spylog.com	Marketing tool SpyLog
194.67.23.251	48	(N/A)	SpyLog too???
207.68.176.190	34	msnsearch.info	MSN Search Engine by Microsoft
208.254.63.69	21	(N/A)	Owned by UUNET INC.

Table 5 Top HTTP Talkers with 46.5.180.250

Hitbox Gateway is a controversial web traffic tracker tool, by WebSideStory, (http://www.websidestory.com/services-solutions/hbp/overview.html), a US company. By actually visiting the web site http://ehg.hitbox.com, I found that it was “Hitbox Gateway 8.7.6 build 3”. When a user access a web page embedded with this tool, the embedded javascript code will collects data and send to this server. According to the company, this is a web site (traffic) analysis tool. But not everybody agrees. CA (Computer Associates) has entry for Hitbox in its “eTrust Spyware Encyclopedia”:

http://www3.ca.com/securityadvisor/pest/pest.aspx?id=453060830

SpyLog is similar to Hitbox, from a Russian company (http://www.spylog.ru/). Unfortunately I do not understand Runssian language. But again, it has an entry in CA's eTrust Spyware Encyclopedia:

http://www3.ca.com/securityadvisor/pest/pest.aspx?id=453060851

With the help of ethereal, I found 1224 among these 4128 packets were HTTP GET requests. Checking the payload of those alerts revealed that the target servers were mostly advertisement servers and some search engines, such as

ad.doubleclick.net	top.list.ru	ads.sina.com.tw	search.aol.com
u001.83.spylog.com	www.topcto.ru	chkpt.zdnet.com	rd.yahoo.com
ad.trafficmp.com	pa.yahoo.com		

Table 6 Partial List of Hosts that 46.5.180.250 accessed through HTTP GET

Using the following rule inside ethereal, I found that at least two HTTP packets contains traces of Gator , a famous adware (http://www.gator.com/, with a new name Claria):

```
http contains Gator
```

This suggests that the host ran a Microsoft Windows system, because Gator can not run on other operating systems. Also, it is known that Microsoft Internet Explorer tends to be much more malware-friendlier than any other web browsers.

For easiness, I used ethereal to analyze the log file. Used filter

```
http && ip.src == 46.5.180.250 && http contains Agent
```

and check the User-Agent value of the HTTP GET requests. We got:

```
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; windows NT 5.0; T312461)\r\n
```

So it is indeed Microsoft Internet Explorer version 5.5 on a Windows 2000 box.

Probability that Source Address was Spoofed

Not likely. The IP address is owned by the University.

Attack Mechanism

(1) CHAT MSN message

In an article by Paul Roberts, “Instant messaging worms pose growing threat”, dated 2003/9/29, at http://www.computerweekly.com/Article125243.htm, a Symantec chief researcher claimed that there were are about 60 published IM vulnerabilities. Although the article did not talk about MSN Messenger only, it did showed us a common situation of several mainstream instant message systems.

According to the article, “... an IM worm could infect 500,000 machines in just 31 seconds”.

Another issue is still privacy issue. The only 2 MSNMS packets captured actually contain the plain text of private chat message. Because the information flow is not encrypted, if people exchange serious information such as password, social security number, etc., the information could be easily intercepted by third parties.

(2) P2P GNUTella client request

Privacy is absolutely a serious concern here, just as we discussed MSN Messenger system. And as stressed, potential legal risks may arise, if appropriate University network usage policy was not developed or enforced.

There have been reported possible exploits around GNUTella, such as

http://www.xatrix.org/print.php?s=726

According to the following article, it is also possible to anonymously launch a DDoS attack via the Gnutella network:

http://www.auscert.org.au/render.html?it=2404&template=1

It has been reported that some viruses can spread through P2P file exchange applications as well, such as this first one found,

http://nts.jhu.edu/alerts/alert.detail.cfm?aid=31

The following table lists IP addresses that 46.5.180.250 sent GNUTella requests to. We can see that they are wildly diversive. This is not surprising at all since it is a P2P protocol.

4.46.225.153	64.15.226.121	66.28.45.164	170.140.186.13
12.248.46.184	64.15.226.122	66.69.194.20	170.140.214.161
24.150.206.243	64.194.106.129	66.70.109.20	172.179.189.6
24.153.17.215	64.83.209.247	66.76.148.194	195.149.55.190
24.159.178.47	65.26.138.117	68.10.72.89	200.13.200.74
24.165.145.34	65.26.51.66	68.0.216.219	200.63.163.250
24.170.5.216	65.35.138.77	68.103.88.236	202.44.41.84
24.241.88.85	65.59.50.65	68.13.146.38	206.186.44.108
24.29.36.31	65.92.30.76	68.3.177.40	212.129.131.88
24.46.221.84	66.169.156.106	80.108.28.13	213.191.87.40
24.50.159.42	66.176.86.29	80.203.20.88	213.93.231.73
24.67.10.215	66.188.168.124	128.192.2.49	216.35.67.146
24.68.232.85	66.222.61.54	141.35.14.47	216.35.73.137
24.84.24.120	66.25.52.132	149.156.118.119	217.134.110.253
64.14.40.168	66.250.52.45	152.19.216.55	217.68.172.130
64.14.40.170	66.28.220.144	157.159.42.75	
64.14.40.178	66.28.45.160	157.181.150.200	
64.14.40.181	66.28.45.162	165.121.88.59	

Table 7 List of IP Addresses 46.5.180.250 Sent GNUTella Requests

(3) HTTP Traffic sent by 46.5.180.250

Malwares such as browser popups and adwares are big annoyances both to normal users and to security analysts. As we have encountered, they generated a lot of false positives for our IDS systems.

Many of them such as Hitbox, Spylog, Gator found in our log, collect surfing information and computer system information of web site visitors. The information was sent to third party instead of the visited web site owner without explicitly informing the users.

"The Spy Swat" discussed details about HitBox in his article "Big Brother: Web – hitbox.com" (http://www.geocities.com/Area51/3543/hitbox.htm).

Some of the modern browsers has ability to defeat this kind of tools in certain level. For example, Mozilla has an option to disable cross-site cookie transfer, which effectively disallow cookies sent to third party's web site. Combining with frustration on some major web browsers' poor security record, many individuals and even organizations have started using alternative browsers such as Mozilla (and its derivatives such as popular Mozilla FireFox). On 2004/12/10, InformationWeek reported "Penn State Tells 80,000 Students To Chuck IE" citing security concern, (http://www.informationweek.com/story/showArticle.jhtml?articleID=55301109). One month later, on 2005/1/18, another report "Some Companies Switching From Microsoft's IE Browser" said that many others followed suite (http://www.informationweek.com/story/showArticle.jhtml?articleID=57701783).

Also note that this kind of HTTP traffic not only raises security/privacy concern for end users of the University, its ability to generate way too many alerts in the IDS system also brings annoyances to network security team.

Evidence of Active Targeting

No.

Severity

(3 + 3) – (2 + 2) = 2

Criticality	3	Many vulnerabilities, some left unfixed
Lethality	3	If infected, we see viruses/worms almost everywhere
System Countermeasures	2	Unknown system level measures; if infected, significant amount of time and effort needed to fix systems
Network Countermeasures	2	Poor filtering on University network border, but IDS detected the attack

Table 8 Severity Assessment for Internal IP 46.5.180.250

Defensive Recommendations

Again, P2P programs have legitimate use, and academic institutions value information sharing. However, to avoid potential legal risks, some policy needs to be developed and enforced. This is a tough job. It is not a pure technical issue, and needs collaboration among various departments/offices of the University, including Legal service.

From a technical point of view, the Network Security Team of IT department has responsibility in assessing the network usage issues and recommend viable options on network infrastructure including hardware, software and users behavior. For example, to reduce the undue traffic generated by many malware, certain alternative products may be better choices then those widely used. If this is the case, IT department should make the recommendations to upper management of the University.

2.5 Generic Defensive Recommendations

I conveyed some defensive recommendations in each detect above. Because not every kinds of detects have been discussed, I would like to stress some of the general points here:

- Review the current University Network Usage Policy and enhance it accordingly
- Initiate Security Awareness program in the University, so that every user is well aware of the situation and familiar with University Network Usage Policy
- Strengthen the University network border's filtering ability, allowing no known exploiting packets coming through it
- Implement different levels of IDS, Such as in DMZ zone and on mission-critical systems
- Implement a security scanning system and policy, locate system vulnerabilities and fix them before bad guys do and exploit them.

3.Analysis Process

3.1 Analysis environment

The whole analysis process was performed entirely on GNU/Linux environment. The hardware system is a PC (P4-HT 2.8GHz and 512 MB RAM) running SuSE Linux 9.2 with the following applications coming with the OS:

- tcpdump version 3.8.3
- libpcap version 0.8.3
- Snort version 2.2.0 (Build 30) with configuration changes (each rule enabled; stream4 preprocessor disabled)
- ethereal version 0.10.6

Additionally, the following software was downloaded from third party and used to generate alert summaries:

- SnortSnarf-021111.1

Many other applications were used for the analysis, such as awk, sed, grep, sort, uniq, etc. Since these are standard UNIX utilities, they are not listed in detail here.

The submission was prepared entirely using OpenOffice.org version 1.1.3 coming with the operating system. The current GIAC Practical Admistrivia accepts only PDF or Microsoft WORD (and compatible) format for submission. This may have made sense two years, but no more. I would rather see that at least OpenOffice.org format be accepted as well. Fortunately, OpenOffice.org allows PDF file exported, so I do not have to switch to Microsoft Windows environment to complete the submission as some other GIAC exam takers did.

3.2 Analysis Steps

- Prepare the necessary software

  1. Although Snort and ethereal packages are included in SuSE distribution, they are not installed by default. So installation and configuration steps are needed.

  2 .SnortSnarf needs to be downloaded from

  http://www.snort.org/dl/contrib/data_analysis/snortsnarf/SnortSnarf-021111.1.tar.gz

  3. Perl module Time::JulianDay needs to be installed to use SnortSnarf.

  perl -MCPAN -e 'install Time::JulianDay'

- Check the file type and find MAC addresses in question
- Find traffic flow information and decide internal network, and draw link graph of the web server and sketch possible network typology graph
- Using Snort and SnortSnarf to generate HTML summaries report

1. Snort needs option “-k none” to turn off checksum verification, as checksums of the packets in the log were scrambled by SANS/GIAC.

2. By default, Snort disabled various rules, such as backdoor.rules, shellcode.rules, p2p.rules, web-attacks.rules, etc. These needs to be enabled by removing commenting sign from the corresponding line in snort.conf.

3. Disable stream4 preprocessor from snort.conf. According to Daniel Wesemann (http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00084.html), stream4 preprocessor needs to be disabled to recover maximum original detects. Jan Stodola discussed briefly the mechanism that stream4 is conterproductive in his GCIA Practical submission (Honor: http://www.giac.org/certified_professionals/practicals/gcia/0754.php).

- Analyze the summary report and look into the log in more detail, and choose detections to describe and for further investigation, using available tools including ethereal. When some outside IP addresses seem to do something malicious or funny, locate them by using SamSpade's whois service at http://www.samspade.org/t/whois.
- Wrap up and finish the paper. OpenOffice.org has been used exclusively for this task.

4.References

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

Stephen Northcutt, Judy Novak
Network Intrusion Detection, Third Edition, New Riders, 2002

Stephen Northcutt, Mark Cooper, Matt Fearnow, Karen Frederick
Intrusion Signatures and Analysis, New Riders, 2001