Global Information Assurance Certification Paper

GIAC Level Two Intrusion Detection In Depth
GCIA Practical Assignment for SANS2002 Orlando, FL
April 1-7 , 2002
Version 3.1

Orazio Mistretta GCFW Advisory Board Member

Assignment 1: Describe the State of Intrusion Detection

What the established operator in a Cisco IOS access list allows in ?

I have found that there is a certain degree of uncertainty with respect to the correct definition of what is allowed by an access list that allows established traffic. In the following paragraphs, there are some examples of what people believe about the word "established", then I will suggest a more correct placement of this statement inside the access list.

From the SANS GCFW Course material [1] we read: "To help solve problems like these, the "established" operator is used. The established operator filters incoming traffic for the existence of either the RST or ACK flags. Theoretically this would allow only established connections, since in accordance with the rules of a standard TCP handshake, a packet making an initial connection would be flagged SYN only.

Established also has a major security flaw. Scanning tools such as nmap have the flexibility to allow you to choose what flags you want your packets to contain.
Hence, established traffic can be "spoofed" using the ACK or RST flags, effectively bypassing a filter using the established keyword… Attackers sometimes use ACKs and they work quite well. If the perimeter defense here is a similar technology to a Cisco router with established set, this scan will penetrate the site. When it reaches a live host computer, that computer will respond with a RST packet and this will let the scanner know about its existence."

Donald Kuntz on his GCFW Practical writes: "The established keyword will allow any tcp packet that has the ACK or RST flag set. With the numerous scanners, which can spoof the ACK and RST flags the established keyword can be easily thwarted, so I normally will not use it." [2]

In another GCFW practical, from Chris Lethaby, we have [3]: "When defining Simple ACL's it was necessary to permit or allow packets based explicitly on source or destination ports. With extended ACL's came the ability to permit traffic based on the 'established' keyword. This evaluated the packet for the presence of the SYN,ACK bits being set, which indicates that a traditional 3-way TCP handshake is occurring, and the subsequent ACK's which appear during the data transfer period of the TP/IP session."

While Gina Montgomery in his GCFW practical [4] writes:

```
access-list 120 permit tcp any 192.216.1.0 0.0.0.63 gt 1023 established
```

allow return connections for TCP sessions initiated from us

Another GCFW graduate, Stephen Carrol in his GCFW practical writes a security policy that states the following:

"! Allow only ACKed tcp packets to our network, to reduce the threat of malicious activity
access-list 101 permit tcp any *any* gt 1023 established" [5]

This is really what matters: we want to reduce the threat of malicious activity, and we use the established operator in an extended access list on a Cisco router with IOS to obtain this goal.

So, what kind of TCP traffic, really, will the established keyword allows in our networks?
Will it allow return traffic for connections initiated from the inside, only?
Will it allow only TCP packets with an ACK or RST on the TCP header byte 13?
And, is it better to avoid the use of this keyword because it offers no additional protection to our perimeter defense?

From the Cisco documentation [6], we don't obtain any additional clues: "The **established** keyword is used only for the TCP protocol to indicate an established connection. A match occurs if the TCP datagram has the ACK or RST bits set, which indicate that the packet belongs to an existing connection."

Remember, TCP is a stateful protocol, and to initiate a conversation, some preliminary packet exchange has to be done, sometimes this is referred to as the "Three way handshake" because it implies a series of three data exchange characterized by the different TCP flag setting on the packet headers:
Client ----------SYN----------→ SERVER
SERVER ------SYN+ACK-----→ Client
Client ----------ACK----------→SERVER

After the completion of this handshake, the two nodes have negotiated the connection and the data transfer can happen in both directions. The tcp handling code of a system behaves like a finite states automata that supports a given number of states that govern the response of the system to incoming packets. The possible responses are better detailed in [7], as a function of the flags present in the arriving packet and the current status of the system.

Hakers have built a long history of using the tcp flag byte (tcp[13] in tcpdump notation) for reconnaissance and OS fingerprinting scopes. Several ready to use tools are available that permit the construction of packets with arbitrary flag settings. Two of the most used of such tools are nmap (authored by Fyodor Yarochin, available at

http://www.insecure.org) and hping2 by Salvatore Sanfilippo (Antirez).

I have found particularly useful for my testing purposes the hping2 tool, that behaves as described in [8]:"hping is a command-line oriented TCP/IP packet assembler/analyzer. The interface is inspired to the ping(8) unix command, but hping isn't only able to send ICMP echo requests".

So I decided to use this tool to perform a series of scans against a system of mine protected by an inbound access list installed on the external interface of a Cisco 2600 router. The aim of this task was to find exactly which packet and which combination of flags passed through an access list like the following:

```
access-list 111 deny   ip host 0.0.0.0 any log
access-list 111 deny   ip 127.0.0.0 0.255.255.255 any log
access-list 111 deny   ip 10.0.0.0 0.255.255.255 any log
access-list 111 deny   ip 172.16.0.0 0.15.255.255 any log
access-list 111 deny   ip 192.168.0.0 0.0.255.255 any log
access-list 111 deny   ip my.net.15.0 0.0.0.255 any log
access-list 111 permit tcp any host my.net.15.3 eq 22
access-list 111 permit tcp any host my.net.15.66 eq smtp
access-list 111 permit tcp any host my.net.15.66 eq 22
access-list 111 permit tcp any host my.net.15.66 eq www
access-list 111 permit tcp host 131.154.1.3 host my.net.15.3 eq domain
access-list 111 deny   tcp any any eq domain log
access-list 111 permit udp any host my.net.15.3 eq domain
access-list 111 permit udp any eq domain any
access-list 111 permit tcp any any established
access-list 111 deny   tcp any any range 0 1024 log
access-list 111 deny   tcp any any range 6000 6064 log
access-list 111 deny   tcp any any range 2000 2064 log
access-list 111 deny   tcp any any eq 2049 log
access-list 111 deny   tcp any any eq 1080 log
access-list 111 deny   tcp any any eq 8080 log
…
access-list 111 deny   tcp any any eq sunrpc log
access-list 111 deny   udp any any log
access-list 111 permit icmp host 193.206.158.30 any echo
access-list 111 permit icmp host 193.206.158.32 any echo
access-list 111 permit icmp any any packet-too-big
access-list 111 permit icmp any any ttl-exceeded
access-list 111 deny   icmp any any log
access-list 111 deny   ip any any log
```

Protected by this access list is a bastion host that runs an instance of tcpdump and collects all the traffic sent to it. Also note the position of the established rules in the access list: it is analyzed before the more restrictive deny rules that block specific ports or port ranges (rpc at tcp/111, nfs at tcp/2049, X at tcp/6000-6064, proxy at tcp/1080 and tcp/8080 and so on).

In a couple of hours, I was able to generate (using hping2 from my Linux box) all the possible sequences with the following combination of flags:

1. only 1 flags set
2. any 2 flags set
3. any 3 flags set
4. any 4 flags set

The result is that the only traffic crossing my established access list is made by packets that contain at least the ACK or the RST flags set, as told in the SANS [1] and in the Cisco [6] documentation. I am looking closer at the behavior of established access list in the Cisco IOS software because I use an access list similar to the previous on my network, but, although I enable access to port 80 only to my web server, my firewall drops incoming packets that the router apparently lets in.
Here is a partial dump of the traces that raised my curiosity:

```
Aug  5 01:54:01 dns screend[494]: REJECT: TCP [24.241.100.210]->[my.net.15.132](2311-
>80)19 (def rule -)
Aug  6 06:11:47 dns screend[496]: REJECT: TCP [218.108.201.166]->[my.net.15.148](3273-
>80)11 (def rule -)
Aug  6 06:13:17 dns screend[496]: REJECT: TCP [218.108.201.166]->[my.net.15.148](3570-
>80)11 (def rule -)
Aug  7 12:02:14 dns screend[496]: REJECT: TCP [62.30.127.30]->[my.net.15.165](3702->80)11
(def rule -)
Aug  8 20:33:04 dns screend[496]: REJECT: TCP [62.30.127.30]->[my.net.15.130](1960->80)11
(def rule -)
Aug  9 03:40:49 dns screend[496]: REJECT: TCP [211.217.235.126]->[my.net.15.133](41405-
>80)16 (rule 1 -)
Aug 17 15:21:07 dns screend[496]: REJECT: TCP [218.54.10.53]->[my.net.15.174](37165->80)04
(def rule -)
Aug 17 15:21:08 dns screend[496]: REJECT: TCP [218.54.10.53]->[my.net.15.174](37198->80)04
(def rule -)
Aug 17 15:21:08 dns screend[496]: REJECT: TCP [218.54.10.53]->[my.net.15.174](37212->80)04
(def rule -)
Aug 17 15:23:06 dns screend[496]: REJECT: TCP [218.54.10.53]->[my.net.15.174](37145->80)15
(def rule -)
Aug 17 15:23:06 dns screend[496]: REJECT: TCP [218.54.10.53]->[my.net.15.174](37148->80)15
(def rule -)
Aug 17 15:23:07 dns screend[496]: REJECT: TCP [218.54.10.53]->[my.net.15.174](37153->80)15
(def rule -)
Aug 17 15:23:07 dns screend[496]: REJECT: TCP [218.54.10.53]->[my.net.15.174](37175->80)15
(def rule -)
Aug 17 15:23:09 dns screend[496]: REJECT: TCP [218.54.10.53]->[my.net.15.174](37198->80)15
(def rule -)
Aug 17 15:23:09 dns screend[496]: REJECT: TCP [218.54.10.53]->[my.net.15.174](37212->80)15
(def rule -)
Aug 19 10:44:43 dns screend[496]: REJECT: TCP [218.54.10.53]->[my.net.15.168](42839->80)15
(def rule -)
Aug 19 10:55:56 dns screend[496]: REJECT: TCP [218.54.10.53]->[my.net.15.162](37288->80)15
(def rule -)
Aug 19 10:55:56 dns screend[496]: REJECT: TCP [218.54.10.53]->[my.net.15.162](37297->80)15
(def rule -)
Sep  7 02:54:33 dns screend[496]: REJECT: TCP [202.96.18.183]->[my.net.15.146](4295->80)04
(def rule -)
Sep  7 02:56:54 dns screend[496]: REJECT: TCP [202.96.18.250]->[my.net.15.159](4664->80)04
(def rule -)
Sep  7 03:01:40 dns screend[496]: REJECT: TCP [202.96.18.250]->[my.net.15.147](1774->80)04
(def rule -)
Sep  7 03:09:17 dns screend[496]: REJECT: TCP [202.96.18.250]->[my.net.15.157](2403->80)04
(def rule -)
Sep  7 03:45:26 dns screend[496]: REJECT: TCP [202.96.18.183]->[my.net.15.166](4736->80)04
(def rule -)
Sep  7 05:17:40 dns screend[496]: REJECT: TCP [202.96.18.250]->[my.net.15.179](2566->80)04
(def rule -)
Sep  7 05:48:38 dns screend[496]: REJECT: TCP [202.96.18.183]->[my.net.15.190](1186->80)04
(def rule -)
Sep  7 05:49:38 dns screend[496]: REJECT: TCP [211.130.76.72]->[my.net.15.143](3484->80)14
(def rule -)
Sep  7 05:55:44 dns screend[496]: REJECT: TCP [202.96.18.183]->[my.net.15.190](2366->80)04
(def rule -)
```

```
Sep  7 06:03:52 dns screend[496]: REJECT: TCP [211.90.139.50]->[my.net.15.135](6030->80)14
(def rule -)
Sep  7 06:21:50 dns screend[496]: REJECT: TCP [211.130.76.72]->[my.net.15.143](1324->80)14
(def rule -)
Sep  7 06:54:49 dns screend[496]: REJECT: TCP [218.10.238.3]->[my.net.15.157](2610->80)04
(def rule -)
Sep  7 07:26:29 dns screend[496]: REJECT: TCP [62.69.66.213]->[my.net.15.171](12682->80)14
(def rule -)
Sep  7 07:40:14 dns screend[496]: REJECT: TCP [211.130.76.72]->[my.net.15.163](1607->80)14
(def rule -)
Sep 11 12:41:26 dns screend[747]: REJECT: TCP [61.10.39.30]->[my.net.15.140](3159->80)19
(def rule -)
Sep 12 15:18:44 dns screend[747]: REJECT: TCP [80.192.218.100]->[my.net.15.177](3115-
>80)11 (def rule -)
Sep 12 20:37:54 dns screend[747]: REJECT: TCP [200.204.199.57]->[my.net.15.156](4135-
>80)19 (def rule -)
Sep 13 15:40:44 dns screend[747]: REJECT: TCP [210.58.1.241]->[my.net.15.132](33385->80)19
(def rule -)
Sep 13 22:59:08 dns screend[747]: REJECT: TCP [62.30.143.234]->[my.net.15.181](4809->80)11
(def rule -)
Sep 14 16:51:55 dns screend[747]: REJECT: TCP [200.204.199.57]->[my.net.15.166](4269-
>80)19 (def rule -)
Sep 14 17:00:31 dns screend[747]: REJECT: TCP [62.31.124.135]->[my.net.15.190](3337->80)11
(def rule -)
Sep 14 17:02:02 dns screend[747]: REJECT: TCP [62.31.124.135]->[my.net.15.190](3082->80)11
(def rule -)
Sep 16 12:22:19 dns screend[747]: REJECT: TCP [200.204.199.57]->[my.net.15.151](1231-
>80)19 (def rule -)
```

Here the screend process in my firewall blocked several connection attempts from the outside to port 80 of a series of hosts that don't exist in my network.
For a better comprehension, it must be told that the screend log format is the following:
Date and time (generated by the syslog daemon)
Hostname and process [PID]:
Type of screend logging rule (ACCEPT, REJECT, REJECTN, SUMMARY)
Protocol [source IP]->[dest IP] (source port)->(dest port)flags (activated rule)
While all other numbers are decimals, the flags byte is reported in hexadecimal.

These packets are bypassing the established access list, and are blocked by the default rule of the screend firewall, because they have one of the following flag patterns:

1. 0x04=0000 0100=RST flag on
2. 0x15=0001 0101=ACK RST FIN flags are on
3. 0x14=0001 0100=ACK RST flags on
4. 0x19=0001 1001=ACK PUSH FIN flags on
5. 0x11=0001 0001=ACK FIN flags on
6. 0x16=0001 0110=ACK RST SYN flags on

Now, while a packet with a lone RST or with RST+ACK or with ACK+FIN looks like a normal packet exchanged in a discussion between two hosts, it should be noted that:

1. these packets are destined to hosts not existing in my network, and are blocked by default by screend
2. they are accompanied by packets with strange flags combination like ACK+RST+FIN or ACK+RST+SYN

Next, I performed a series of checks to establish what kind of responses two different systems generated, if hit by different flag combinations on open and on closed ports. The two system I had the possibility to test are a Windows 2000 server and a Tru64 UNIX system.

I used hping2 to hit them with the flag combinations in the tcp header reported in the following paragraphs.

Response of Windows 2000 Server

OS	Win2k	Win2k	Win2k	Win2k	Win2k	Win2k	Win2k	Win2k	Win2k	Win2k
FLAGS	ACK	SYN	FIN	PUSH	URG	RST	ACK SYN FIN	FIN ACK	FIN ACK RST	FIN RST ACK URG
Listen Port	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *
No Listen Port	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **
Risp. Stack TCP/IP OS	RST * RST **	SYN ACK * RST ACK **	RST ACK * RST ACK **	RST ACK * RST ACK **	RST ACK * RST ACK **	NOR * **	RST * RST **	RST * RST **	NOR * **	NOR * **

OS	Win2k	Win2k	Win2k	Win2k	Win2k	Win2k	Win2k	Win2k	Win2k	Win2k
FLAGS	RST SYN	SYN ACK	SYN ACK RST	SYN FIN RST URG PUSH ACK	NULL	PUSH URG	SYN PUSH	SYN URG		
Listen Port	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *
No Listen Port	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **
Risp. Stack TCP/IP OS	NOR * **	RST * RST **	NOR * **	NOR * **	RST ACK * RST ACK **	RST ACK * RST ACK	SYN ACK * RST ACK **	SYN ACK * RST ACK **		

Response of Tru64 v.5.0

OS	Tru64	Tru64	Tru64	Tru64	Tru64	Tru64	Tru64	Tru64	Tru64	Tru64
FLAGS	ACK	SYN	FIN	PUSH	URG	RST	ACK SYN FIN	FIN ACK	FIN ACK RST	FIN RST ACK URG
Listen Port	22/tcp *	22/tcp *	22/tcp *	22/tcp *	22/tcp *	22/tcp *	22/tcp *	22/tcp *	22/tcp *	22/tcp *

No Listen Port	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **
Risp. Stack TCP/IP OS	RST * RST **	SYN ACK * RST ACK **	NOR * RST ACK **	NOR * RST ACK **	NOR * RST ACK **	NOR * NOR **	RST * RST **	RST * RST **	NOR * NOR **	NOR * NOR **

OS	Tru64	Tru64	Tru64	Tru64	Tru64	Tru64	Tru64	Tru64	Tru64	Tru64
FLAGS	RST SYN	SYN ACK	SYN ACK RST	SYN FIN RST URG PUSH ACK	NULL	PUSH URG	SYN PUSH	SYN URG		
Listen Port	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *	139/tcp *
No Listen Port	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **	80/tcp **
Risp. Stack TCP/IP OS	NOR * NOR **	RST * RST **	NOR * NOR **	NOR * NOR **	NOR * RST ACK **	NOR * RST ACK **	SYN ACK * RST ACK **	SYN ACK * RST ACK **		

In constructing these tables, I used a W2000 server which has an tcp/139 active port and a non-listening port at tcp/80. The Unix system had a listening application (sshd2) at port tcp/22, and the non-listening port is tcp/80. The NOR is an abbreviation for no-response to the injected packet. The following hping2 options were used to generate the packets:

```
-F
     set FIN tcp flag.

-S
     set SYN tcp flag.

-R
     set RST tcp flag.

-P
     set PUSH tcp flag.

-A
     set ACK tcp flag.

-U
     set URG tcp flag.

-X
     set Xmas tcp flag.

-Y
     set Ymas tcp flag.
```

Conclusions

The established access list of the Cisco IOS based routers permits to limit the kind of packets that reach our hosts. Fundamentally, it permits to control the traffic originated from outside that belongs to tcp connections initiated from our internal network. However, given that the IOS is not a stateful firewall implementation (only the reflexive access lists offer a kind of stateful behaviour), one cannot completely trust them.

Forged packets with a combination of flags but with at least the ACK or the RST flag set will penetrate the perimeter. The response of the target system depends on the tcp/ip stack of the manufacturer.

Given that the access list rules are examined in the order they have been declared in the configuration file, and because the rules scrutiny finishes as soon as a match is found between the arriving packet and each single rule, it becomes apparent that the position of the established rule inside the access list is crucial to provide a better degree of protection.

For this reason, the best place to put the established rule, in the access list chain, is immediately before a deny ip any any log (block all not explicitly permitted traffic) and after the explicit deny rules that block all the most dangerous ports that we don't want expose to risk, as is done in the following example:

```
access-list 111 deny   ip host 0.0.0.0 any log
access-list 111 deny   ip 127.0.0.0 0.255.255.255 any log
access-list 111 deny   ip 10.0.0.0 0.255.255.255 any log
access-list 111 deny   ip 172.16.0.0 0.15.255.255 any log
access-list 111 deny   ip 192.168.0.0 0.0.255.255 any log
access-list 111 deny   ip my.net.15.0 0.0.0.255 any log
access-list 111 permit tcp any host my.net.15.3 eq 22
access-list 111 permit tcp any host my.net.15.66 eq smtp
access-list 111 permit tcp any host my.net.15.66 eq 22
access-list 111 permit tcp any host my.net.15.66 eq www
access-list 111 permit tcp host 131.154.1.3 host my.net.15.3 eq domain
access-list 111 deny   tcp any any eq domain log
access-list 111 permit udp any host my.net.15.3 eq domain
access-list 111 permit udp any eq domain any
access-list 111 deny   tcp any any range 0 1024 log
access-list 111 deny   tcp any any range 6000 6064 log
access-list 111 deny   tcp any any range 2000 2064 log
access-list 111 deny   tcp any any eq 2049 log
access-list 111 deny   tcp any any eq 1080 log
access-list 111 deny   tcp any any eq 8080 log
…
access-list 111 deny   tcp any any eq sunrpc log
access-list 111 deny   udp any any log
access-list 111 permit icmp host 193.206.158.30 any echo
access-list 111 permit icmp host 193.206.158.32 any echo
access-list 111 permit icmp any any packet-too-big
access-list 111 permit icmp any any ttl-exceeded
access-list 111 deny   icmp any any log
access-list 111 permit tcp any any established
access-list 111 deny   ip any any log
```

This implementation offers more protection because we can drop all the external packets

addressed toward dangerous open services on the internal hosts, and log all the denied connections. In this way we reach at least two objectives:

1. we don't disclose the service structure of our internal hosts, except for the official servers/services we offer to the Internet community
2. we can log the access trials directed at sensitive services that are good candidates for misuse

However, it should be noted an access list on the border router is only the first barrier against the outsiders that try to penetrate our perimeter. The best defence mechanism is based on the "defence in depth principle" that suggests to deploy several layers of defensive mechanisms: routers, firewalls, IDS. Last, but not least, all the internal hosts must be fortified against malicious activities with the installation of all the relevant security patches and the removal of all unneeded services.

References

[1] Defense In-Depth Day 3: SANS' Track Two Perimeter Protection Defense In-Depth, SANS Institute

[2] http://www.giac.org/practical/Donald_Kuntz.doc

[3] http://www.giac.org/practical/Chris_Lethaby_GCFW.zip

[4] http://www.giac.org/practical/Gina_Montgomery_GCFW.zip

[5] http://www.giac.org/practical/Stephen_Carroll_GCFW.zip

[6] http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fipr_c/ipcprt1/1cfip.htm

[7] http://webmail.cotse.com/CIE/RFC/793/34.htm

[8] http://www.hping.org

Assignment 2 – Network Detects

2.1 Skitter traces

2.1.1 Source of Trace

These events were collected after changing the network configuration at a customer site. Before the 23 of May, we had a simple network, where the external router had an IP address of my.net.15.254 and the web server was at IP address my.net.15.3.

Starting from May 23 2002, we installed a firewall, installed a central log server, configured a series of access lists on a new external router (Cisco 2600 with IOS v12.2), and assigned to all internal machines IANA reserved address in the subnets 192.168.1.0/24 and 192.168.5.0/24.

In a couple of hours we started to collect on the syslog server the following traces:

```
May 24 22:51:08 router 18087: May 24 22:51:30.509: %SEC-6-IPACCESSLOGDP: list 111 denied
icmp 204.29.239.23 -> my.net.15.254 (8/0), 1 packet
May 25 09:06:02 router 24022: May 25 09:06:24.026: %SEC-6-IPACCESSLOGDP: list 111 denied
icmp 204.29.239.23 -> my.net.15.3 (8/0), 1 packet
```

Several days later, the traces were even more interesting:

```
Jun 15 16:49:37 router 252509: Jun 15 16:49:13.363: %SEC-6-IPACCESSLOGDP: list 111 denied
icmp 204.29.239.23 -> my.net.15.3 (8/0), 1 packet
Jun 15 16:53:29 router 252527: Jun 15 16:53:05.383: %SEC-6-IPACCESSLOGDP: list 111 denied
icmp 204.29.239.23 -> my.net.15.3 (8/0), 17 packets
Jun 16 08:25:35 router 257666: Jun 16 08:25:09.811: %SEC-6-IPACCESSLOGDP: list 111 denied
icmp 204.29.239.23 -> my.net.15.254 (8/0), 1 packet
Jun 16 08:28:29 router 257685: Jun 16 08:28:04.579: %SEC-6-IPACCESSLOGDP: list 111 denied
icmp 204.29.239.23 -> my.net.15.254 (8/0), 17 packet
```

A system with IP 204.29.239.23 was pinging (?) our old router and web server addresses and the access list on the router was blocking this activity. Careful examination of the log files revealed that this was not the only source address generating this type of scan (?), but there were other systems whose activity was perfectly similar (1 packet followed by 17 ping packets directed only to our old router address and to our old web server address).

2.1.2 Detect was generated by

The source of the trace is the log of an access list activated on the border router of the customer network. It is an input access list that screens the packets coming from the Internet connection.
An extract of the router config is given to clarify better the situation:

```
...
interface Serial0/0
 bandwidth 2048
 ip address myext.ip.137.162 255.255.255.252
 ip access-group 111 in
 no ip redirects
 no ip unreachables
```

```
 no ip proxy-arp
 no ip mroute-cache
 load-interval 30
 no cdp enable
…
access-list 111 deny   ip host 0.0.0.0 any log
access-list 111 deny   ip 127.0.0.0 0.255.255.255 any log
access-list 111 deny   ip 10.0.0.0 0.255.255.255 any log
access-list 111 deny   ip 172.16.0.0 0.15.255.255 any log
access-list 111 deny   ip 192.168.0.0 0.0.255.255 any log
access-list 111 deny   ip my.net.15.0 0.0.0.255 any log
access-list 111 permit tcp any host my.net.15.3 eq 22
access-list 111 permit tcp any host my.net.15.66 eq smtp
access-list 111 permit tcp any host my.net.15.66 eq 22
access-list 111 permit tcp any host my.net.15.66 eq www
access-list 111 permit tcp host 131.154.1.3 host my.net.15.3 eq domain
access-list 111 deny   tcp any any eq domain log
access-list 111 permit udp any host my.net.15.3 eq domain
access-list 111 permit udp any eq domain any
access-list 111 deny   tcp any any range 0 1024 log
access-list 111 deny   tcp any any range 6000 6064 log
access-list 111 deny   tcp any any range 2000 2064 log
access-list 111 deny   tcp any any eq 2049 log
access-list 111 deny   tcp any any eq 1080 log
access-list 111 deny   tcp any any eq 8080 log
access-list 111 deny   udp any any eq 2049 log
access-list 111 deny   udp any any eq netbios-ns log
access-list 111 deny   udp any any eq netbios-dgm log
access-list 111 deny   udp any any eq netbios-ss log
access-list 111 deny   udp any any eq tftp log
access-list 111 deny   udp any any eq sunrpc log
access-list 111 deny   tcp any any eq sunrpc log
access-list 111 deny   udp any any log
access-list 111 permit icmp host 193.206.158.30 any echo
access-list 111 permit icmp host 193.206.158.32 any echo
access-list 111 permit icmp any any packet-too-big
access-list 111 permit icmp any any ttl-exceeded
access-list 111 deny   icmp any any log
access-list 111 permit tcp any any established
access-list 111 deny   ip any any log
```

The rule that generated the traces is reported in bold. We reject all the ICMP datagrams that come from the external interface of the router with a few exceptions:

1. ping packets coming from the management stations of the GARR
2. packet too long (fragmentation needed) coming from everywhere
3. ttl exceed coming from everywhere (we would like to traceroute external Ips from the firewall)

2.1.3 Probability the source address was spoofed:

The hosts generating these packets should not be spoofed, otherwise they will not receive the answer (echo reply). It is not a DOS attack, given that the time interval between successive trials is very high (hours), most of the incoming packets are from registered hosts, so the source address should be the real one.

2.1.4 Description of attack:

The traces shown in paragraph 2.1.1 were generated by a single external IP: 204.29.239.23. Since we regularly analyze the log files generated by our router/firewall/IDS that we collect on the syslog server, we observed that several other hosts were pinging the two

addresses at IP my.net.15.254 and my.net.15.3.
While my.net.15.254 is no longer used, it was the address of the external router before we changed the network config. Furthermore, my.net.15.3 while once it was the IP address of the web server, now is the address of the firewall itself!

Utilizing a grep on the collection of all the ICMP records logged, I found all the systems that were sending ICMP 8/0 (echo request, ping/traceroute) to my.net.15.254. All these systems are sending packets to my.net.15.3, too. All these systems manifest the same behaviour: 1 packet to each destination, followed by 17 packets to each destination.

```
Jul 25 18:54:49 router 649915: Jul 25 18:53:01.444: %SEC-6-IPACCESSLOGDP: list 111 denied
icmp 192.172.226.24 -> my.net.15.254 (8/0), 1 packet
Jul 25 18:57:51 router 649936: Jul 25 18:56:03.180: %SEC-6-IPACCESSLOGDP: list 111 denied
icmp 192.172.226.24 -> my.net.15.254 (8/0), 17 packets
Jul 25 19:57:16 router 650400: Jul 25 19:55:29.076: %SEC-6-IPACCESSLOGDP: list 111 denied
icmp 205.189.33.78 -> my.net.15.254 (8/0), 1 packet
Jul 25 19:58:14 router 650440: Jul 25 19:56:27.300: %SEC-6-IPACCESSLOGDP: list 111 denied
icmp 205.189.33.78 -> my.net.15.254 (8/0), 17 packets
```

In a couple of month we received this kind of packets from the following IPs:
(I used the unix command
`grep my.net.15.254 icmp |grep "(8/0)"| awk '{ print $14 }' |sort |uniq -c |sort –rn`
to obtain the list. The icmp file contains all the records that we logged because they contain an icmp packet blocked by the external router.)

```
Count Source IP
=================
  117 205.189.33.78
  114 209.249.139.254
   72 192.172.226.24
   64 204.29.239.23
   50 128.223.162.38
    2 209.48.138.47
    2 208.50.214.101
```

Trying to find the reason of this behavior, I did a search with nslookup to try to obtain the host names of these machines. I found the following :
205.189.33.78 is yto.skitter.caida.org
209.249.139.254 is nrt.skitter.caida.org
192.172.226.24 is riesling-ether.caida.org
204.29.239.23 is mw.skitter.caida.org
128.223.162.38 is uoregon.skitter.caida.org
209.48.138.47 is testweb3.sqa.sockeye.com
208.50.214.101 is not defined in the DNS

With these data on the table, it looks like that a company that owns the caida.org domain is generating the Events of Interest (EOI) activity against my network. Such activity was surely started long before I installed the firewall and the new router (with the access list) and is going on.

I am excluding from my analysis the last two IP address because they really were

generating network detects with a different pattern: they don't show the repetition of 1 and 17 packets and they probe also other systems in my.net.

2.1.5 Attack mechanism:

The detects we are analyzing are part of a coordinated reconnaissance effort. The ICMP type 8 code 0 packets can be generated by a host using the ping command or by a windows host using the tracert (traceroute) command.

2.1.6 Correlations

"CAIDA, the Cooperative Association for Internet Data Analysis, provides tools and analyses promoting the engineering and maintenance of a robust, scalable global Internet infrastructure." Caida is based at the University of California San Diego Supercomputing Center.
A look at the Caida web site gives an explanation of what kind of events we are receiving: http://www.caida.org/tools/measurement/skitter/. Skitter is the software they developed to record "each hop from a source to many destinations. by incrementing the "time to live" (TTL) of each IP packet header and recording replies from each router (or hop) leading to the destination host" with the following scope: "CAIDA's measurement efforts are intended to help users, providers and researchers understand the complexities in the current and future Internet. skitter research will provide the community with insight into the complexity of a large, heterogeneous, and dynamic worldwide topology."
Other security conscious people were concerned by skitter probing and reported signs of skitter activities in the following:
http://www.giac.org/practical/Roland_Gerlach_GCIA.html
http://www.daemonnews.org/199902/news.html
http://archives.neohapsis.com/archives/linux/suse/2001-q1/0048.html

2.1.7 Evidence of active targeting

Only two systems are probed by skitter: the old my.net router and the old web server. No broadcast addressees are present in the records, no other systems are probed by the CAIDA monitors. It is a focused probing against my.net.15.254 and my.net.15.3.

2.1.8 Severity

We measure the severity of these traces according to the following formula, taken from the Intrusion Detection in Depth, track 3, SANS Institute:
severity = (criticality + lethality) – (system countermeasures + network countermeasures)
where each value is ranked on a scale from 1 (lowest) to 5 (highest).
In this case we have:
Criticality: 5 (target are our firewall and Internet router)

Lethality: 1 (information gathering is not lethal)
System countermeasures: 5
The router is protected by his access lists, the firewall is a hardened bastion host. Logging and alarming are performed on a syslog host isolated from all other systems.
Network countermeasures: 5
Access list on the border router and the firewall should block all unauthorized traffic.
SEVERITY= (5+1)-(5+5)=-4

2.1.9 Defensive recommendation

Defenses worked fine. The reconnaissance attack was blocked at the perimeter and generated a log that raised an alarm. Security people were able to trace the intent of the perpetrators.
An email can be sent to skitter-configs@caida.org asking them to discontinue their measurements against our addresses.

2.1.10 Multiple choice test question

In the following trace, what kind of activity your border router is preventing to enter your network?

```
Jun 16 08:28:29 router 257685: Jun 16 08:28:04.579: %SEC-6-IPACCESSLOGDP: list 111 denied
icmp 204.29.239.23 -> my.net.15.254 (8/0), 17 packet
```

a. ICMP echo reply packets
b. ICMP redirect packets
c. ICMP echo request packets

Response: C

2.2 Connection to Port 20480

```
Sep 1 03:22:56 dns screend[496]: SUMMARY:4 of TCP [192.168.1.13]->[207.188.7.131](34309-
>20480)
Sep 1 06:07:05 dns screend[496]: SUMMARY:4 of TCP [192.168.1.13]->[207.188.7.131](38405-
>20480)
Sep 1 08:54:51 dns screend[496]: SUMMARY:4 of TCP [192.168.1.13]->[207.188.7.131](42501-
>20480)
Sep 1 11:57:15 dns screend[496]: SUMMARY:4 of TCP [192.168.1.13]->[207.188.7.131](55814-
>20480)
Sep 1 13:26:33 dns screend[496]: SUMMARY:4 of TCP [192.168.1.13]->[207.188.7.131](34055-
>20480)
Sep 1 15:51:20 dns screend[496]: SUMMARY:4 of TCP [192.168.1.13]->[207.188.24.156](516-
>20480)
Sep 1 16:14:04 dns screend[496]: SUMMARY:4 of TCP [192.168.1.13]->[207.188.24.156](1028-
>20480)
Sep 1 16:16:09 dns screend[496]: SUMMARY:8 of TCP [192.168.1.13]->[64.12.180.19](1540-
>20480)
Sep 1 16:20:01 dns screend[496]: SUMMARY:4 of TCP [192.168.1.13]->[64.12.180.22](2564-
>20480)
```

```
Sep 1 16:32:01 dns screend[496]: SUMMARY:4 of TCP [192.168.1.13]->[207.188.24.156](772-
>20480)
Sep 1 16:33:00 dns screend[496]: SUMMARY:4 of TCP [192.168.1.13]->[207.188.24.156](1540-
>20480)
Sep 1 20:34:47 dns screend[496]: SUMMARY:4 of TCP [192.168.1.13]->[207.188.7.131](62216-
>20480)
```

The detects shown here are only a sample of a series of EOI we are gathering because a number of internal hosts that try to connect directly to external servers on, apparently, tcp port 20480.

2.2.1 Source of Trace

The traces have been generated on a customer network protected by a bastion host where we installed a firewall based on the SCREEND packet filter.
SCREEND is perhaps one of the first packet filters developed in the world. It was the result of a research effort made at the NSL Laboratory of Digital Equipment Corporation. It was written by Jeffrey Mogul of DEC and now is part of the Tru64 operating system distributed by HP/Compaq. For a complete description of screend, its configuration file and its limitations, please, look at http://research.compaq.com/nsl/publications/TN-2.html

The site where these EOI have been gathered has a strict security policy that states that each connection attempt to external systems should pass through a proxy on the firewall. No direct connections are allowed between internal clients and external systems. In addition, the internal users are not allowed to install arbitrary software on their systems, to prevent copyright infringement and legal issues.

2.2.2 Detect was generated by

The operation of screend is driven by a configuration file that defines, in an orderly fashion, the actions to be performed on each packet. When the gateway receives a packet that should be forwarded, instead of passing it to the forwarding kernel module, it invokes the gwscreen kernel module that calls the screend daemon. Screend analyzes the packet starting with the first rule in its config file and stops only when a match is found. If the packet matches a rule, the action taken on it (forward or reject) depends on the rule itself, otherwise the so called default rule is invoked. In our network the default rule rejects all packets.

The screend config file that we use is the following:

```
for my.net.15.0 netmask is 255.255.255.192;
for my.net.15.64 netmask is 255.255.255.192;
for my.net.15.128 netmask is 255.255.255.192;
for 192.168.1.0 netmask is 255.255.255.0;
for 192.168.5.0 netmask is 255.255.255.0;
# Block SF and SR and SP and SU scans
from host any to host any tcp port any flags syn fin reject log;
from host any to host any tcp port any flags syn rst reject log;
from host any to host any tcp port any flags syn push reject log;
from host any to host any tcp port any flags syn urg reject log;
# Block XMAS scans
from host any to host any tcp port any flags fin push urg reject log;
# Block NULL scans
```

```
from host any to host any tcp port any flags-not syn fin push urg ack rst reject log;
#
between  any and host my.net.15.66  tcp port 80 accept;
between  any and host my.net.15.66  tcp port 25 accept;
between  host my.net.15.66 and any  tcp port 25 accept;
between subnet 192.168.1.0 and host my.net.15.66 tcp port 23  accept;
between subnet 192.168.1.0 and host my.net.15.66 tcp port 25  accept;
between subnet 192.168.1.0 and host my.net.15.66 tcp port 110  accept;
between subnet 192.168.5.0 and host my.net.15.66 tcp port 23  accept;
between subnet 192.168.5.0 and host my.net.15.66 tcp port 25  accept;
between subnet 192.168.5.0 and host my.net.15.66 tcp port 110  accept;
between host 192.168.1.33 and host my.net.15.66 accept;
#
between any udp port domain and subnet my.net.15.128 accept;
#
# from any to any udp port whod reject notify;
# from any to any udp port timed reject notify;
# Default rule
default reject log;
```

Screend normally generates a log record for each packet for which there is a rules with an action of type "log". In our config file we have a default rule that drops all the packets that didn't match any other rule, and logs the packet.
The log format is the following:
Date and time (generated by the syslog daemon)
Hostname and process [PID]:
Type of screend log (ACCEPT, REJECT, REJECTN, SUMMARY)
Protocol [source IP]->[dest IP] (source port)->(dest port)

So the following record

```
Sep 1 03:22:56 dns screend[496]: SUMMARY:4 of TCP [192.168.1.13]->[207.188.7.131](34309->20480)
```

means: screend process 496 on host dns screened 4 TCP packets going from host 192.168.1.13:34309 to host 207.188.7.131:20480.

The SUMMARY records of screend do not say anything about the packet disposition: in our case all these packets were dropped (rejected) without further notice to the source host (no-notify).

2.2.3 Probability the source address was spoofed

Here we see packets coming from an internal system (a PC), an antispoof filter protects the firewall from accepting invalid packets from all the interfaces.
The ifaccess (anti-spoof mechanism of Tru64) config file we use is the following:

```
ee0 127.0.0.1 255.255.255.255 denylog
ee0 0.0.0.0  255.255.255.255 denylog
ee0 10.0.0.0  255.0.0.0 denylog
ee0 172.16.0.0  255.240.0.0 denylog
ee0 192.168.0.0  255.255.0.0 denylog
tu0 127.0.0.1 255.255.255.255 denylog
tu0 my.net.15.66 255.255.255.255 permit
tu0 0.0.0.0 0.0.0.0 denylog
tu1 127.0.0.1 255.255.255.255 denylog
tu1 192.168.1.0 255.255.255.0 permit
tu1 my.net.15.128 255.255.255.192 permit
tu1 0.0.0.0 0.0.0.0 denylog
```

```
ee1 127.0.0.1 255.255.255.255 denylog
ee1 192.168.5.0 255.255.255.0 permit
ee1 0.0.0.0 0.0.0.0 denylog
```

Here eeo, ee1, tu0 and tu1 are the firewall interfaces. The only interface that accepts packets from the 192.168.1.0 network is tu1, that connects all the Pc's located in the lecture hall. So the addresses are true, no spoofing is present.

2.2.4 Description of attack

These packets raised my attention because they violate the security policy of an educational institution were I installed the screend based firewall. A group of internal hosts (there are more records in our logs that I can show in the traces) is trying to access an external server on TCP port 20480. This could be the sign of a possible compromise of the PC's, or the side effect of a user installing not licensed software on it.
The target server 207.188.7.131 belongs to Real Networks Inc. (dns name: oupostrr1.real.com), the company that authored the RealPlayer software.
A look at the arin whois database reveals:

```
#arin 207.188.24.156

OrgName:    RealNetworks, Inc.
OrgID:      REAL

NetRange:   207.188.0.0 - 207.188.31.255
CIDR:       207.188.0.0/19
NetName:    PROGNET-REAL
NetHandle:  NET-207-188-0-0-1
Parent:     NET-207-0-0-0-0
NetType:    Direct Allocation
NameServer: BENNY.PROGNET.COM
NameServer: RUGBUG.PROGNET.COM
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    1999-02-22
Updated:    2001-06-20

TechHandle: IR57-ARIN
TechName:   RealNetworks, Inc.
TechPhone:  +1-206-892-6737
TechEmail:  net-admin@real.com
```

The address: 64.12.180.19 corresponds to main-v3.netscape.com while the address 64.12.180.22 is main-v4.netscape.aol.com.
So our internal systems are probing some big software company servers at port 20480 for some unknown reason.

2.2.5 Attack mechanism

Initially, I was very confused about these traces interpretation. A search on the web looking for tcp port 20480 gave me a suggestion. At http://lists.paco.net/oops-eng/msg00139.html I found the suggestion that a bug in the code could produce a wrong port number: instead of port 80 we can find port 20480, the result of shifting the value 80 of one byte to the left (20480=80*256).

This scenario is more realistic: an internal system tries to access external web server at

Real.com or at netscape.com without using the proxy server at the firewall and the screend blocks the packets. But who is the source of the false interpretation? And what application is responsible for not using the proxy located at 192.168.1.1:8080 that I configured in the Internet properties of each PC?
Trying to find an answer to these questions I reached the following conclusions:

1. the PC showing this activity were all running a copy of RealPlayer
2. they run also a copy of Netscape browser and AOL IM
3. using the command
   ```
   tcpdump –i tu1 'src host 192.168.1.11 and dst host 207.188.7.131'
   ```
   I found that the PC generated regularly packets going to this external host but addressed to port 80, not port 20480
4. I used also tcpdump to trace other source host-dest host conversations that showed similar logs (SUMMARY screend log of tcp packet with dest port 20480) and found only packets directed to external server tcp port 80

So I reached the following conclusions:

1. internal users installed some software on their PC's
2. There is a bug in the log.c module of the screend source distributed by HP/Compaq. This error produces a wrong representation of the destination port number only when SUMMARY record are generated. When screend generates a log record for each REJECT(ED) packet, the error is not present.
3. A Software Performance Report will be generated and sent to HP/Compaq, asking them to correct the error.

2.2.6 Correlations

No correlations have been found. Screend is not a widely deployed firewall (packet filter) software, and perhaps those who use it have not seen these strange messages in the log files. In any case I have not seen a message indicating possible misinterpretations of the tcp header by screend.

Since I use screend as a firewall tool in other places (including my company networks), I looked at the log files generated on these systems and I found other instances of these kind of events (internal PC's accessing an external web server bypassing the proxy and filtered by screend with a wrong tcp destination port of 20480).

2.2.7 Evidence of active targeting

The traces are produced by client software authored by Netscape and Real Networks. These packages try to connect to their producer's web server to verify the availability of updates, or to display the default home page, or to connect to a music channel.

2.2.8 Severity

In this case we have:

Criticality: 2 (target is ouside our network, but an internal PC could be compromised)
Lethality: 4 (Probing of external systems misinterpreted by the firewall)
System countermeasures: 2 (No integrity check or software inventory solution is installed on the PC)
Network countermeasures: 5
The firewall blocked the packets, but reported a wrong port number, further raising my attention
SEVERITY= (2+4)-(2+5)=-1

2.2.9 Defensive recommendation

Defenses worked fine: direct connection between internal systems and external servers is blocked by the packet filter. I suggest to improve the control of the internal systems with a software capable of performing an integrity check of the file systems and/or an inventory of the software packages installed. The possibility to avoid that a user installs arbitrary software without any control could be enforced with a solution like Computer Associates' AIM/IT (Asset and Inventory Management) or similar.

A request has been fired to HP/Compaq to correct the bug in screend.

2.2.10 Multiple choice test question

Trace:

```
Sep 1 03:22:56 dns screend[496]: SUMMARY:4 of TCP [192.168.1.13]->[207.188.7.131](34309->20480)
```

Question:
Which of the following IDS/Firewall software generated the trace?

1. Cisco PIX
2. Ascend Router
3. Enterasys Dragon IDS
4. SCREEND
5. Snort 1.8.x
6. ISS RealSecure

Answer: 4

2.3 Formmail

2.3.1 Source of Trace

The source of the following traces is an Sicilian educational institution that runs a Netscape FasTrack/3.01 Web server. The server is located behind a firewall that allows only web and mail access to this host.
The target host runs sendmail 8.12.4 as the mail server of the organization.

No other services are enabled to external users by the access control list on the router and firewall combination.
Here are the detects generated by an IDS system based on snort 1.8.4 and the Web server itself (they were sanitized not to show the real IP addresses of the customer network and to not disclose detailed info wrt domain/host names):

```
09/07-19:07:15.460672  [**] [1:884:2] WEB-CGI formmail access [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 61.56.198.68:31671 -> my.net.15.66:80
09/07-19:08:08.514352  [**] [1:884:2] WEB-CGI formmail access [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 61.56.198.68:31955 -> my.net.15.66:80
09/07-19:08:09.035136  [**] [1:884:2] WEB-CGI formmail access [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 61.56.198.68:31955 -> my.net.15.66:80
09/08-16:10:54.726144  [**] [1:884:2] WEB-CGI formmail access [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 67.242.141.62:4161 -> my.net.15.66:80
09/09-11:22:46.253760  [**] [1:884:2] WEB-CGI formmail access [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 68.129.23.54:1630 -> my.net.15.66:80
09/10-06:46:58.702720  [**] [1:884:2] WEB-CGI formmail access [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 63.67.170.95:39800 -> my.net.15.66:80
09/10-06:47:39.263520  [**] [1:884:2] WEB-CGI formmail access [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 63.67.170.95:40120 -> my.net.15.66:80
09/10-06:47:47.150304  [**] [1:884:2] WEB-CGI formmail access [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 63.67.170.95:40214 -> my.net.15.66:80
09/10-08:09:48.668560  [**] [1:884:2] WEB-CGI formmail access [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 4.63.173.73:1538 -> my.net.15.66:80
```

Here several hosts are looking for the presence of the formmail cgi script on the web server. The access file on the Web server reports the following (the time difference is due to the fact that the systems are not synchronized and no NTP server is in place):

```
s068.diginet.com.tw - - [07/Sep/2002:19:00:51 +0100] "POST /cgi-bin/formmail.pl HTTP/1.0"
404 310
s068.diginet.com.tw - - [07/Sep/2002:19:00:40 +0100] "POST /cgi-bin/formmail.pl HTTP/1.0"
404 306
s068.diginet.com.tw - - [07/Sep/2002:19:01:45 +0100] "POST /cgi-bin/formmail.pl HTTP/1.0"
404 306
1Cust62.tnt14.dca5.da.uu.net - - [08/Sep/2002:16:04:36 +0100] "GET /cgi-
bin/formmail.pl?email=lafam&subject=www%2Emydom%2Ecompany%2Eit%2Fcgi%2Dbi
n%2Fformmail%2Epl&recipient=dizzo%40email%2Enu&msg=Formmail_Found! HTTP/1.1Content-Type:
application/x-www-form-urlencoded" 404 207
1Cust54.tnt1.tucson.az.da.uu.net - - [09/Sep/2002:11:16:27 +0100] "GET /cgi-
bin/formmail.pl?email=f2%40aol%2Ecom&subject=www%2Emydom%2Ecompany%2E
it%2Fcgi%2Dbin%2Fformmail%2Epl&recipient=cardingn0p1%40aol%2Ecom&msg=w00t HTTP/1.1Content-
Type: application/x-www-form-urlencoded" 404 207
wap.usana.com - - [10/Sep/2002:06:40:40 +0100] "POST /cgi-bin/formmail.pl HTTP/1.0" 404
342
wap.usana.com - - [10/Sep/2002:06:41:23 +0100] "POST /cgi-bin/formmail.pl HTTP/1.0" 404
310
wap.usana.com - - [10/Sep/2002:06:41:15 +0100] "POST /cgi-bin/formmail.pl HTTP/1.0" 404
305
tamqfl1-ar2-4-63-173-073.tamqfl1.dsl-verizon.net - - [10/Sep/2002:08:03:25 +0100] "GET
/cgi-bin/formmail.pl?email=f2%40aol%2Ecom&subject=www%2
Emydom%2Ecompany%2Eit%2Fcgi%2Dbin%2Fformmail%2Epl&recipient=applec5%40aol%2Ecom&msg=w00t
HTTP/1.1Content-Type: application/x-www-form-urlencoded" 404 207
```

2.3.2 Detect was generated by

The original detect was generated by the following snort rule:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI formmail access";flags: A+;
uricontent:"/formmail"; nocase; reference:bugtraq,1187; reference:cve,CVE-1999-0172;
reference:arachnids,226; classtype:attempted-recon; sid:884; rev:2;)
```

This rule generates an alert when it detects a TCP packet coming from any TCP port of any external system and destined to one of the internal HTTP_SERVERS on tcp port 80 containing an URI with the string "/formmail".
This detect triggered our attention, so we looked at the access and error file of the web server.

2.3.3 Probability the source address was spoofed

The source addresses reported by the traces are the real attacking hosts, because the snort rule activates after a successful three way handshake has completed the TCP connection phase (flags A+ in the snort signature).
The attacking hosts are:
Name: s068.diginet.com.tw
Address: 61.56.198.68
Name: 1Cust62.tnt14.dca5.da.uu.net
Address: 67.242.141.62
Name: 1Cust54.tnt1.tucson.az.da.uu.net
Address: 68.129.23.54
Name: wap.usana.com
Address: 63.67.170.95
Name: tamqfl1-ar2-4-63-173-073.tamqfl1.dsl-verizon.net
Address: 4.63.173.73

2.3.4 Description of attack

Ronal F. Guilmette and Justin Mason made an excellent work in describing various vulnerabilities of several versions (up to version 1.9) of FormMail , a perl script authored by Matt Wright. Their work is available at: http://www.monkeys.com/anti-spam/formmail-advisory.pdf
The attacks shown in the traces are not the exploitation of any particular vulnerabilty of this script itself, they are scan attempts looking for a vulnerable server, whose address is sent to the hacker mothership (a mail basket of an AOL user in most of cases).
The latest version of formmail (1.91) has been produced with particular attention to the removal of various vulnerabilities, see the details at: http://www.scriptarchive.com/readme/formmail.html

There are 5 CVE entries or candidates relating to FormMail.
CVE version: 20020625

Name	Description
CVE-1999-0172	FormMail CGI program allows remote execution of commands.
CVE-1999-0173	FormMail CGI program can be used by web servers other than the host server that the program resides on.

CVE-2000-0255	The Nbase-Xyplex EdgeBlaster router allows remote attackers to cause a denial of service via a scan for the FormMail CGI program.
CVE-2000-0411	Matt Wright's FormMail CGI script allows remote attackers to obtain environmental variables via the env_report parameter.
CAN-2001-0357	FormMail.pl in FormMail 1.6 and earlier allows a remote attacker to send anonymous email (spam) by modifying the recipient and message paramaters.

2.3.5 Attack mechanism

The packets coming from the IP 67.242.141.62, 68.129.23.54 and 4.63.173.73 all share a similar behavior: they try to send a forged email using formmail on the attacked web server. The email is sent to someone who collects lists of hosts that utilize formmail. These hosts could be utilized to send anonymously unsolicited email (spam).
While the email field is the user return address that appears in the email message, the subject of the email gives the name of the target host and the recipient field is the email address of the hacker who launches the scan and collects the target data.
In my case we have:
User return address: lafam, f2@aol.com
Recipients: dizzo@email.nu, applec5@aol.com
Messages: Formmail_Found!, w00t
Subjects: www.mydom.company.it/cgi-bin/formmail.pl

```
1Cust62.tnt14.dca5.da.uu.net - - [08/Sep/2002:16:04:36 +0100] "GET /cgi-
bin/formmail.pl?email=lafam&subject=www%2Emydom%2Ecompany%2Eit%2Fcgi%2Dbi
n%2Fformmail%2Epl&recipient=dizzo%40email%2Enu&msg=Formmail_Found! HTTP/1.1Content-Type:
application/x-www-form-urlencoded" 404 207

1Cust54.tnt1.tucson.az.da.uu.net - - [09/Sep/2002:11:16:27 +0100] "GET /cgi-
bin/formmail.pl?email=f2%40aol%2Ecom&subject=www%2Emydom%2Ecompany%2E
it%2Fcgi%2Dbin%2Fformmail%2Epl&recipient=cardingn0p1%40aol%2Ecom&msg=w00t

tamqfl1-ar2-4-63-173-073.tamqfl1.dsl-verizon.net - - [10/Sep/2002:08:03:25 +0100] "GET
/cgi-bin/formmail.pl?email=f2%40aol%2Ecom&subject=www%2
Emydom%2Ecompany%2Eit%2Fcgi%2Dbin%2Fformmail%2Epl&recipient=applec5%40aol%2Ecom&msg=w00t
```

Grepping a little on the web server access log, I found that in the last year over 491 formmail attacks were received, 197 contained the word w00t as the message text, while only 58 had the “Formmail_Found!” message string. Whenever the message contained the user return address of f2@aol.com (198 times), it had also (197 times) a message text of w00t.

```
# grep formmail access | cut -d \? -f2 | cut -d \& -f1 | grep email | sort | uniq -c |
sort -rn
 198 email=f2%40aol%2Ecom
  20 email=lafam
  11 email=Skanned%40aol%2Ecom
   2 email=rockstar@mail.com
   1 email=xxx0@xvq34.com
   1 email=rbb56@fff61.com
   1 email=meh47@pkz2.com
   1 email=gqg78@raq10.com
   1 email=formmailed%40yahoo%2Ecom
   1 email=blt45@jmk30.com
```

```
   1 email=bii68@ehi96.com
```

The recipients list we collected is the following:

```
# grep formmail access | cut -d \? -f2 | cut -d \& -f1 | grep recipient |sort|uniq -c
|sort -rn
16 recipient=billybook@telkom.net
   8 recipient=elulis%40aol%2Ecom
   8 recipient=applec5%40aol%2Ecom
   6 recipient=sexbuggyblue@aol.com,
   6 recipient=jxffrey%40aol%2Ecom
   6 recipient=cardingn0p1%40aol%2Ecom
   5 recipient=rmseeds@aol.com
   5 recipient=a2%40myrealbox%2Ecom
   5 recipient=VlRGIN%40aol%2Ecom
   4 recipient=sup%40deepfriedfilms%2Ecom
   4 recipient=lkjasdf@aol.com
   4 recipient=joejoe@aol.com
   4 recipient=jillkillian@acmemail.net
   4 recipient=formmailrogers@yahoo.com
   4 recipient=briansbling%40aol%2Ecom
   3 recipient=tcatenaccio%40mail%2Ecom
   3 recipient=mustangloversvo@aol.com
   3 recipient=loonydreamz%40yahoo%2Ecom
   3 recipient=griffkrn@aol.com
   3 recipient=elulis2%40aol%2Ecom
   3 recipient=dizzo%40email%2Enu
   3 recipient=cardingn0p%40aol%2Ecom
   3 recipient=bulkmarketer@aol.com
   3 recipient=black@value.net
   3 recipient=allpremo%40aol%2Ecom
   2 recipient=unavaiiable%40aol%2Ecom
   2 recipient=tequilagb%40yahoo%2Ecom
   2 recipient=talley@mail.com
   2 recipient=t00hipp0rk@aol.com
   2 recipient=superwedgie415@aol.com
   2 recipient=shinkeys%40aol%2Ecom
   2 recipient=sexbuggyblue@aol.com
   2 recipient=roflolmfao%40yahoo%2Ecom
   2 recipient=openrelayspy@flashmail.com
   2 recipient=mugs10%40aol%2Ecom
   2 recipient=mousetray2k@aol.com
   2 recipient=magicman00152@aol.com,
   2 recipient=lrs1034%40aol%2Ecom
   2 recipient=lexus430ls%40aolc%2Eom
   2 recipient=laysan%40aol%2Ecom
   2 recipient=kineticenergy%40boxfrog%2Ecom
   2 recipient=juney99us@yahoo.com
   2 recipient=horrormovies%40aol%2Ecom
   2 recipient=dek%40epimp%2Ecom
   2 recipient=dave@djtasq.com,tasq@djtasq.com,blah@djtasq.com,root@djtasq.com
   2 recipient=danisahoe%40yahoo%2Ecom
   2 recipient=bluesky2114@aol.com
   2 recipient=binary%40nyc%2Err%2Ecom
   2 recipient=barbutos01%40comcast%2Enet
   2 recipient=antioxidant%40aol%2Ecom
   2 recipient=acewild%40emailaccount%2Ecom
   2 recipient=ZoSpamalams%40aol%2Ecom
   2 recipient=MikeYaDeadHommie%40aol%2Ecom
   2 recipient=
   1 recipient=zzteknose%40aol%2Ecom
   1 recipient=zipptie@aol.com
   1 recipient=youlosefats%40cs%2Ecom
   1 recipient=xxn20xx%40epimp%2Ecom
   1 recipient=xck%40aol%2Ecom
   1 recipient=whoadank%40aol%2Ecom
   1 recipient=vo1o%40aol%2Ecom
```

```
1 recipient=vandsauto%40aol%2Ecom
1 recipient=userpolox%40aol%2Ecom
1 recipient=unconfidable%40aol%2Ecom
1 recipient=travismiiler%40aol%2Ecom
1 recipient=tinaluvslx%40aol%2Ecom
1 recipient=tim121012%40yahoo%2Ecom
1 recipient=thcdavidzkn%40aol%2Ecom
1 recipient=term1nal%40msn%2Ecom
1 recipient=tbwhite@flashmail.com
1 recipient=tashaye13%40aol%2Ecom
1 recipient=syrinyx%40msn%2Ecom
1 recipient=syco%40epimp%2Ecom
1 recipient=stackjob%40aol%2Ecom
1 recipient=soyboidsl%40aol%2Ecom
1 recipient=smackdown2002l%40yahoo%2Ecom
1 recipient=slicka%40epimp%2Ecom
1 recipient=sizam%40epimp%2Ecom
1 recipient=sire%5F2k2%40hotmail%2Ecom
1 recipient=shinko%40epimp%2Ecom
1 recipient=secretimprtntmsg%40aol%2Ecom
1 recipient=se7enisheaven%40aol%2Ecom
1 recipient=scanz00%40aol%2Ecom
1 recipient=sanderz217%40aol%2Ecom
1 recipient=sacto420%40epimp%2Ecom
1 recipient=sacramentoca420%40aol%2Ecom
1 recipient=ruggedchlld1983%40aol%2Ecom
1 recipient=romo303x%40rediffmail%2Ecom
1 recipient=rmitchell9601@aol.com,
1 recipient=ripbooofcbay%40aol%2Ecom
1 recipient=remedialmath%40aol%2Ecom
1 recipient=reign%40epimp%2Ecom
1 recipient=qpsfor200%40aol%2Ecom
1 recipient=pvill412%40aol%2Ecom
1 recipient=piaygreg%40aol%2Ecom
1 recipient=palmviisunday%40aol%2Ecom
1 recipient=ouwop%5Ficonz%40epimp%2Ecom
1 recipient=origionai%40aol%2Ecom
1 recipient=oenmike%40aol%2Ecom
1 recipient=nightmonkey420%40aol%2Ecom
1 recipient=mrbadkitty5%40hotmail%2Ecom,mrbadkitty6%40hotmail%2Ecom
1 recipient=mk2turntables%40aol%2Ecom
1 recipient=mikeyrulzeDamike@aol.com
1 recipient=miacp16%40aol%2Ecom
1 recipient=methud%40hotmail%2Ecom
1 recipient=marcyadams12@hotpop.com
1 recipient=lrs1033%40aol%2Ecom
1 recipient=lildizzay%40aol%2Ecom
1 recipient=leet2k1%40epimp%2Ecom
1 recipient=laoboys%5Fbankn%40hotmail%2Ecom
1 recipient=kripystanks%40aol%2Ecom
1 recipient=kimbanger59@aol.com
1 recipient=kevin2kz%40aol%2Ecom
1 recipient=kengak%40aol%2Ecom
1 recipient=kelly18sexycoed%40netscape%2Ecom
1 recipient=jordan%40subtech%2Enet
1 recipient=jonnystyle%40aol%2Ecom
1 recipient=jhgjkh@mh.com
1 recipient=jazzz003%40aol%2Ecom
1 recipient=jamesway%40aol%2Ecom
1 recipient=j0n%40freeze%2Ecom
1 recipient=itzpainter%40aol%2Ecom
1 recipient=itsmemiked%40hotmail%2Ecom
1 recipient=irayden%40hotmail%2Ecom
1 recipient=impeaching%40aol%2Ecom
1 recipient=illrunuover%40yahoo%2Ecom
1 recipient=ibeanzl%40aol%2Ecom
1 recipient=iastxleft%40aol%2Ecom
```

```
1 recipient=hogporn%40yahoo%2Ecom
1 recipient=hogporn%40hotmail%2Ecom
1 recipient=hemplants%40yahoo%2Ecom
1 recipient=hemperature%40aol%2Ecom
1 recipient=h0llabackladys%40aol%2Ecom
1 recipient=gtxcash%40aol%2Ecom
1 recipient=gregfarkel@aol.com
1 recipient=gregdogg19%40aol%2Ecom
1 recipient=goodslife187%40aol%2Ecom
1 recipient=formmails4me%40epimp%2Ecom
1 recipient=formmailmist%40aol%2Ecom
1 recipient=formmailed%40yahoo%2Ecom
1 recipient=formcooltest123%40hotmail%2Ecom
1 recipient=form1957@aol.com
1 recipient=fockingflyboi%40aol%2Ecom
1 recipient=flooidzzz%40aol%2Ecom
1 recipient=fithy04%40aol%2Ecom
1 recipient=fadex101%40aol%2Ecom
1 recipient=f2%40epimp%2Ecom
1 recipient=enveemysoul%40email%2Ecom
1 recipient=dre%40email%2Ecom
1 recipient=dooshkms%40aol%2Ecom
1 recipient=dispose%40aol%2Ecom
1 recipient=declple%40aol%2Ecom
1 recipient=cstrikeEdog%40aol%2Ecom
1 recipient=cronost51@aol.com
1 recipient=crlsz40@aol.com
1 recipient=compaqbling%40aol%2Ecom
1 recipient=chewmama69@aol.com
1 recipient=cgimaiier%40aol%2Ecom
1 recipient=cgibiz%40aol%2Ecom
1 recipient=cfh@aol.com
1 recipient=castmc@aol.com
1 recipient=cacacoco83%40hotmail%2Ecom
1 recipient=c0ol2%40hotmail%2Ecom
1 recipient=buggzy29@aol.com
1 recipient=bornaking%40hotmail%2Ecom
1 recipient=bmreazy%40aol%2Ecom
1 recipient=bigdoaf%40aol%2Ecom
1 recipient=bannester%40aol%2Ecom
1 recipient=backupsnz4flyboi%40aol%2Ecom
1 recipient=ashepooh2@aol.com
1 recipient=arson%5Fan%5Ftekneo%40hotmail%2Ecom
1 recipient=aoerect%40aol%2Ecom
1 recipient=agenthemp%40aol%2Ecom
1 recipient=acetrace%40emailaccount%2Ecom
1 recipient=accentual%40aol%2Ecom
1 recipient=a8s8%40aol%2Ecom
1 recipient=TruRule%40aol%2Ecom
1 recipient=TopNotchmark@aol.com
1 recipient=Spamminthang%40aol%2Ecom
1 recipient=SpamFormmails%40aol%2Ecom
1 recipient=Soyboidsl%40aol%2Ecom
1 recipient=Sneakerking2k2%40aol%2Ecom
1 recipient=Sekc spammed you@aol.com
1 recipient=SM0KINL0TUS%40AOL%2ECOM
1 recipient=QUEST%40aol%2Ecom
1 recipient=PoliceBeater%40aol%2Ecom
1 recipient=Lrs1034%40aol%2Ecom
1 recipient=IIsodaII%40aol%2Ecom
1 recipient=I%20b%20trickdaddy%40aol%2Ecom
1 recipient=HLGHTLMES%40AOL%2ECOM
1 recipient=GiantKungFuRobot%40aol%2Ecom
1 recipient=Catscrubby%40aol%2Ecom
1 recipient=<Major%20drug%20deals%40aol%2Ecom>formmailed@yahoo.com
```

2.3.6 Correlations:

Similar data have been reported in
http://www.impressions.com.my/watch/annual/2002/01/01/cgi.html
and in
http://www.silicondefense.com/pipermail/snortsnarf-users/2002-March/000248.html and in http://mail.python.org/pipermail/zope/2002-May/115422.html
where they reported the w00t message string and the usage of a return address like f2@aol.com or applec5@aol.com

I found other 2 GIAC practicals reporting traces of formmail probing/exploitation: these can be found at:
http://www.giac.org/practical/Bente_Petersen_GCIA.doc
http://www.giac.org/practical/Dan_Hawrylkiw_GCIA.doc

2.3.7 Evidence of active targeting

Since this site policy of accepting web traffic only directed toward the official web server is enforced by the router access list and the presence of a firewall bastion host, we don't see other formmail accesses other than those directed toward our web server. Perhaps, among all the TCP port 80 packets that we block at the perimeter some contain the same traces and are directed toward other internal IPs. If this is true, we are victim of a random (?) scan that tries to identify potential targets to utilize for more critical activities, otherwise I should suspect that some kind of reconnaissance has been done previously to identify the web server.

2.3.8 Severity

In this case we have:
Criticality: 5 (target is the Web/Mail server of the Institution)
Lethality: 4 (spamming could expose the customer to legal problems and/or insertion into the RBL blacklist)
System countermeasures: 4 (There are no cgi scripts, nor a cgi directory in the web server, but the host is also a mail server)
Network countermeasures: 5
Acces list on the border router and the firewall should block all unauthorized traffic.
SEVERITY= (5+4)-(4+5)=0

2.3.9 Defensive recommendation

The site is well protected, only a web server is directly exposed to Internet users. No cgi scripts are running on the web server, no cgi-bin directory exists.

Unfortunately this host is also running the company mail server. A better solution would be to separate these critical functions and use two separate systems for web and mail services.

2.3.10 Multiple choice test question

Trace:

```
09/07-19:07:15.460672  [**] [1:884:2] WEB-CGI formmail access [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 61.56.198.68:31671 -> my.net.15.66:80
```

Question:

Which of the following statements is true?

1. This trace is activated by a TCP packet directed to a vulnerable mail server
2. This trace alerts on possible vulnerabilities of the IIS web server
3. This trace is a snort alert triggered by a formmail ISAPI buffer overflow
4. This trace is raised by a packet containing a formmail attack to a web server

Answer: 4

Assignment 3 – Analyze This

Executive Summary

The following report contains an analysis of intrusion attempts that occurred at the University in a seven day period in June 2002. The aim of the report is to provide the University security team with a clear picture of the behavior of the internal network, the services offered to the outside world, the attacks perpetrated by internal and/or external users, the evidence of possible system compromise, the signs of network and/or server misconfiguration.

The first step of the analysis process was to develop a summary of all the alerts generated and to extract the top 10 list of source host and destination host and port.
The next step of the analysis process was to develop a picture of the University network, identifying the routers, servers and the high value target hosts that could represent preferred targets of an attack. The network profile was built from the attacks that were logged against each host, in particular looking at established connections and/or particular responses from each host.

To summarize the report, we suggest that a more complete and sound analysis be devoted to the following hosts that showed signs of possible compromise:

1. 130.85.5.19, 130.85.5.88, 130.85.70.177, 130.85.5.42, 130.85.151.90 and 130.85.88.245 by SubSeven
2. 130.85.151.90 by myserver, NIMDA, SubSeven and Red Worm
3. a large number of systems on the subnets 130.85.152.0/24, 130.85.153.0/24 and 130.85.6.0/24 possibly host the Red Worm Trojan
4. 130.85.157.250 by NIMDA

Furthermore, to avoid to expose the internal systems, a more restrictive inbound filter should be installed on the external router/firewall. This filter should allow unrestricted access to the official services that the University deliberately offers to the Internet community, while it should block all other inbound packets that are not part of a session started by internal systems.

Finally, it is suggested to install an anti-virus software on the POP servers to avoid to spread the infection to the client population.

Files analyzed

To accomplish this assignment, I have chosen to analyze the following data taken from http://www.intrusion.org/logs :

Alert.020611.gz	Oos_Jun.11.2002.gz	Scans.020611.gz
Alert.020612.gz	Oos_Jun.12.2002.gz	Scans.020612.gz

Alert.020613.gz	Oos_Jun.13.2002.gz	Scans.020613.gz
Alert.020614.gz	Oos_Jun.14.2002.gz	Scans.020614.gz
Alert.020615.gz	Oos_Jun.15.2002.gz	Scans.020615.gz
Alert.020616.gz		Scans.020616.gz
Alert.020617.gz	Oos_Jun.17.2002.gz	Scans.020617.gz

Total alert records : 862607
Total scan records : 2333373
Total oos records: 227

Alert File Analysis

All snort alerts are reported here with the number of occurrences of each alert. I report in bold all the home made alert definitions, i.e. all the alerts whose definition is not included in the standard snort distribution of signatures. These are signatures developed by the University security team, or are signatures that changed their definition between different versions of snort.

```
Count  Alert Message (Description)
================================
65275  SMB Name Wildcard
55549  SNMP public access
52003  spp_http_decode: IIS Unicode attack detected
30203  ICMP Echo Request L3retriever Ping
28301  MISC Large UDP Packet
21999  INFO Possible IRC Access
10179  INFO MSN IM Chat data
6344  AFS - Off-campus activity
4139  INFO Inbound GNUTella Connect request
4060  ICMP Echo Request Nmap or HPING2
3898  High port 65535 udp - possible Red Worm - traffic
2995  suspicious host traffic
2419  spp_http_decode: CGI Null Byte attack detected
2408  WEB-MISC Attempt to execute cmd
2268  FTP DoS ftpd globbing
2163  Watchlist 000220 IL-ISDNNET-990517
1298  ICMP Fragment Reassembly Time Exceeded
1162  ICMP Router Selection
 713  WEB-IIS view source via translate header
 562  INFO Outbound GNUTella Connect request
 522  Incomplete Packet Fragments Discarded
 439  ICMP Destination Unreachable (Communication Administratively
Prohibited)
 345  Null scan!
 288  ICMP Destination Unreachable (Host Unreachable)
 284  IDS552/web-iis_IIS ISAPI Overflow ida nosize
 268  SCAN Proxy attempt
 246  ICMP Echo Request Windows
 215  NIMDA - Attempt to execute cmd from campus host
 107  ICMP Echo Request CyberKit 2.2 Windows
 105  INFO - Possible Squid Scan
  92  WEB-IIS _vti_inf access
```

```
87  WEB-FRONTPAGE _vti_rpc access
80  IRC evil - running XDCC
77  ICMP traceroute
73  SUNRPC highport access!
71  INFO FTP anonymous FTP
65  Possible trojan server activity
54  ICMP Echo Request BSDtype
53  WEB-MISC http directory traversal
45  WEB-MISC 403 Forbidden
39  WEB-IIS Unicode2.pl script (File permission canonicalization
38  NMAP TCP ping!
38  INFO Napster Client Data
32  ICMP Destination Unreachable (Protocol Unreachable)
31  Watchlist 000222 NET-NCFC
28  WEB-IIS Unauthorized IP Access Attempt
27  ICMP Echo Request Delphi-Piette Windows
26  Attempted Sun RPC high port access
22  UDP SRC and DST outside network
21  WEB-CGI scriptalias access
20  WEB-MISC compaq nsight directory traversal
20  High port 65535 tcp - possible Red Worm - traffic
14  EXPLOIT x86 setuid 0
14  EXPLOIT x86 NOOP
12  Back Orifice
11  SCAN Synscan Portscan ID 19104
11  SCAN FIN
10  Queso fingerprint
10  EXPLOIT x86 setgid 0
 9  EXPLOIT NTPDX buffer overflow
 8  INFO Inbound GNUTella Connect accept
 6  MISC traceroute
 5  WEB-CGI redirect access
 4  Port 55850 udp - Possible myserver activity - ref. 010313-1
 3  Port 55850 tcp - Possible myserver activity - ref. 010313-1
 3  MISC source port 53 to <1024
 3  MISC PCAnywhere Startup
 2  Virus - Possible scr Worm
 2  Virus - Possible pif Worm
 2  TFTP - Internal UDP connection to external tftp server
 1  X11 outgoing
 1  WEB-CGI formmail access
 1  Virus - Possible MyRomeo Worm
 1  TFTP - External UDP connection to internal tftp server
 1  SCAN XMAS
 1  Probable NMAP fingerprint attempt
 1  INFO Outbound GNUTella Connect accept
 1  EXPLOIT x86 stealth noop
```

The alerts can be further subdivided into several categories based on the severity of the attack, in particular:

1. High severity: indicate internal host compromise or the presence of Trojan, virus, backdoor on an internal system
2. Medium severity: attacks aimed to perform reconnaissance activity like ping sweep, post scans, OS fingerprint, banner grabbing, host enumeration
3. Low severity: include internal ping, status monitoring, network browsing

TOP ten lists

These lists were obtained counting the number of records generated by each IP in the alert file generated concatenating all the 7 days alerts.

```
TOP 10 destination IP:
==> tallies.dstips.log <==
32673 130.85.11.7
28411 130.85.150.195
27196 130.85.11.6
13103 130.85.153.157
7828 66.28.132.168
7517 66.62.70.248
6657 64.246.34.181
5876 130.85.88.140
5094 130.85.150.84
4799 130.85.5.96

TOP 10 destination ports:
==> tallies.dstports.log <==
65275 137 (Netbios/SMB access)
55551 161 (SNMP protocol query)
22072 6667 (IRC access)
12693 1336 (IS Chat)
6345 7001 (AFS)
6213 80 (WWW)
5792 2469
4981 1863 (MSNP, Messenger Protocol)
4397 6346 (Gnutella)
3569 2259

TOP 10 source IP:
==> tallies.srcips.log <==
22103 130.85.151.90
17259 130.85.11.7
14249 130.85.11.6
13583 130.85.70.177
12708 140.142.8.72
9117 130.85.88.181
6864 130.85.5.89
5814 202.102.249.118
5381 130.85.150.198
4684 130.85.88.203
```


Network Servers

FTP Servers:
FTP servers were identified using the following command:

```
grep '> 130.85.*:21$' alert |awk ' { print $(NF)}'|cut -d \: -f1|sort|uniq -c|sort -rn
```

Note that all the FTP packets addressed to internal destinations are stimuli and not responses: there are no FTP alert packets generated by internal hosts (source IP in the

130.85.0.0/16 network with source port 21)

The FTP servers are:

```
Count IP address
==================
 699 130.85.153.179
 372 130.85.153.197
 256 130.85.153.180
 188 130.85.152.186
 119 130.85.150.241
 101 130.85.153.191
  96 130.85.88.223
  79 130.85.153.157
  65 130.85.153.162
  62 130.85.150.46
  60 130.85.153.210
  57 130.85.88.163
  34 130.85.151.109
  32 130.85.151.70
  27 130.85.151.14
  16 130.85.153.168
  12 130.85.88.187
  10 130.85.88.171
  10 130.85.150.195
   5 130.85.5.92
   5 130.85.105.120
   4 130.85.150.147
   3 130.85.151.114
   3 130.85.150.83
   3 130.85.150.231
   2 130.85.5.95
   2 130.85.153.219
   2 130.85.150.84
   2 130.85.150.243
   2 130.85.150.228
   2 130.85.150.226
   2 130.85.150.16
   2 130.85.150.143
   2 130.85.150.107
   2 130.85.150.101
   1 130.85.5.137
```

SSH servers:
no ssh/ssd2 servers were identified in the alert files

Telnet Servers:
Even if the command

```
grep '130.85.*:23 ->' alert | cut -d "]" -f3 | cut -d ">" -f1 | cut -d \: -f1 | sort |
uniq -c | sort -rn
```

gave the following result:

```
Count IP address
==============
  15  130.85.186.17
  11  130.85.186.16
```

I believe that there is no telnet server because a manual scrutiny of these records reveals that these are invalid packets: all are NULL scans, even if they look like responses from a telnet server to an external stimulus.

MAIL servers:

no mail/smtp traffic has been logged in the alert files

DNS servers:
The alert file reveals possible dns servers at 130.85.1.2 and 130.85.137.7. The scan file analysis, however doesn't confirm this hypothesis: dns servers generate responses to all the clients that ask for host name-ip translation and thus the aggregated traffic toward them is very high in terms of conversations. This traffic could be intercepted by the snort portscan preprocessor if it is configured with a default trigger of 4 connections in a 3 seconds time interval.

Inspected with this consideration, the scan files revealed other possible dns servers at 130.85.1.3, 130.85.1.4, 130.85.1.5: we looked at all the internal hosts scanned on port 53 by a number of internal/external hosts greater that 100.

```
grep '\:53 UDP' scans | cut -d " " -f 6 | cut -d \: -f1 | sort | uniq -c | sort -rn
Count Dest IP address
=====================
61025 130.85.1.3
33094 130.85.1.4
 2910 130.85.1.5
   23 168.126.63.1
   14 130.85.88.245
    4 130.85.153.168
```

Of course, the there is a big difference between the real DNS server and other systems, scanned on port 53. The figures contained in the accumulated scan file also reveal that the preferred dns server is at IP 130.85.1.3 (first declaration in the /etc/resolv.conf file), followed by 130.85.1.4 and 130.85.1.5.

This hypothesis is confirmed by a nslookup of the umbc.edu domain:

```
Server:  umbc5.umbc.edu
Address:  130.85.1.5

umbc.edu
 origin = UMBC3.umbc.edu

…
umbc.edu     nameserver = UMBC5.umbc.edu
umbc.edu     nameserver = UMBC3.umbc.edu
umbc.edu     nameserver = UMBC4.umbc.edu
…
UMBC3.umbc.edu internet address = 130.85.1.3
UMBC4.umbc.edu internet address = 130.85.1.4
UMBC5.umbc.edu internet address = 130.85.1.5
```

Web Servers were identified using the following commands:

```
grep "> 130.85.*:80" alert | grep -v 8080|cut -d "]" -f3 | cut -d ">" -f2 |cut -d \: -
f1|sed s/\ //g | sort | uniq -c | sort -rn
and
grep '130.85.*:80 ->' alert | cut -d "]" -f3 | cut -d ">" -f1 | cut -d \: -f1|sort | uniq -
c | sort -rn
```

In this way I was looking for stimuli or responses coming from internal web servers.
They are:

```
Count IP address
==================
1425 130.85.5.96
 573 130.85.5.97
 552 130.85.5.95
 477 130.85.5.92
 475 130.85.150.220
 464 130.85.150.246
 453 130.85.150.143
 426 130.85.150.101
 219 130.85.150.6
 167 130.85.153.219
 136 130.85.88.187
 136 130.85.150.63
 136 130.85.150.228
 128 130.85.150.243
 122 130.85.153.73
 110 130.85.150.147
  95 130.85.88.171
  90 130.85.150.83
  82 130.85.150.133
  81 130.85.150.195
  79 130.85.151.114
  69 130.85.150.84
  63 130.85.150.231
  62 130.85.150.226
  58 130.85.88.156
  58 130.85.5.249
  57 130.85.150.107
  34 130.85.150.16
  20 130.85.150.113
  12 130.85.157.248
   8 130.85.150.59
   6 130.85.150.51
   4 130.85.5.44
   3 130.85.5.108
   2 130.85.153.127
   2 130.85.11.4
   1 130.85.5.67
   1 130.85.153.145
   1 130.85.153.108
   1 130.85.152.17
   1 130.85.151.77
   1 130.85.150.245
   1 130.85.111.140
```

Some of these systems appear to be managed switches/routers because they are regularly probed with SNMP queries by the management stations. They are: 130.85.88.187, 130.85.150.195, 130.85.150.84, 130.85.150.147, 130.85.153.219, 130.85.5.108.

POP2 servers were identified using the following command:

```
grep '130.85.*:110 ->' alert
```

They are: 130.85.6.39 and 130.85.6.7

NTP servers:
NTP server candidates (UDP and TCP 123) were identified using the following command:

```
grep '130.85.*:123$' alert | cut -d "]" -f3 | cut -d ">" -f2 | cut -d \: -f1 | sort | uniq -
c | sort -rn

Count IP address
```

```
====================
   6  130.85.88.245
   1  130.85.153.165
   1  130.85.153.145
   1  130.85.152.216
```

However, the NTPDX buffer overflow is triggered by a udp packet, so we have no confirmation that the system is running the NTPD process. A more precise NTP server identification can be obtained using the scans data because these servers generate many responses to all clients, that could be intercepted by the snort portscan preprocessor if the default trigger of 4 connections in a time interval of 3 seconds is used. As a matter of facts, we have:

```
# grep '\:123 \-\>' scans | cut -d " " -f 4 | sort | uniq -c | sort -rn
356227 130.85.60.43:123
   980 130.85.6.45:123
   385 130.85.149.33:123
   360 130.85.149.35:123
   280 130.85.149.38:123
   277 130.85.149.15:123
   162 130.85.149.30:123
   123 130.85.149.34:123
   104 130.85.149.56:123
    90 130.85.149.49:123
    77 130.85.149.60:123
```

All the NTP hosts in the 130.85.149.0/24 subnet exhibit the same behavior because they generate packets directed only toward the hosts 3, 4, 5, 8, 9 and 10 of the subnet 130.85.1.0/24. The hosts 130.85.60.43 and 130.85.6.45 serve only the subnet 153.

SNMP mgmt stations (>100 SNMP packets):

```
Count  Source IP address      Unique destinations   Top target
=================================================================
13370  130.85.70.177          28                    130.85.5.97
 9051  130.85.88.181          2                     130.85.150.84
 5375  130.85.150.198         105                   130.85.113.202
 4569  130.85.88.203          1                     130.85.150.195
 4516  130.85.88.159          1                     130.85.150.195
 4497  130.85.88.207          1                     130.85.150.195
 4484  130.85.88.145          1                     130.85.150.195
 4477  130.85.88.136          1                     130.85.150.195
 2413  130.85.150.245         1                     130.85.152.109
  729  130.85.88.251          1                     130.85.150.195
  504  130.85.88.212          1                     130.85.150.195
  466  130.85.88.225          1                     130.85.150.195
  358  130.85.153.178         1                     130.85.150.147
  345  130.85.91.74           17                    130.85.150.178
  151  130.85.186.10          1                     130.85.153.219
  127  130.85.153.191         1                     130.85.150.147
```

Routers:
The following command was used to derive the routers that responded with the ICMP unreachable (Communication Administratively Prohibited) messages:

```
grep Prohibited alert | cut -d "]" -f3 | cut -d ">" -f1 | sort | uniq -c | sort -nr
Count Source IP
==================
 353  130.85.150.1
```

```
  76  130.85.16.25
   9  130.85.5.1
```

while the ICMP host unreachable messages permitted to identify other routers:

```
grep 'Host Unreach' alert|cut -d \] -f3|cut -d ">" -f1|sort|uniq -c|grep 130.85|sort -rn

Count Source IP
====================
  24  130.85.184.1
  22  130.85.85.1
  22  130.85.3.1
  21  130.85.105.1
  20  130.85.15.1
  18  130.85.140.1
  17  130.85.180.1
  17  130.85.145.1
  17  130.85.115.1
  14  130.85.110.1
  11  130.85.150.1
  10  130.85.70.1
   9  130.85.182.1
   9  130.85.165.1
   9  130.85.100.1
   8  130.85.181.1
   8  130.85.17.1
   8  130.85.162.1
   6  130.85.6.1
   5  130.85.190.1
   2  130.85.168.1
   2  130.85.16.25
   2  130.85.16.14
   2  130.85.141.1
   1  130.85.130.1
```

The following list contains the most frequent alerts (top 10 in terms of number of EOI generated) that are subject of a detailed analysis:

1. SMB Name Wildcard Severity:low

The snort rule that triggered these alerts looks like:

```
alert udp any any -> $HOME_NET 137 (msg:"SMB Name Wildcard";  content:
"CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA|0000|";)
```

No external host is involved in these network traces. Almost a half of all the detects are generated by the following hosts:

Source IP	No. of detects
130.85.11.7	15368
130.85.11.6,	12858
130.85.5.89	4198
130.85.11.5.	1591
130.85.152.172	1110

This traffic can be generated by the command nbtstat –A <dest-IP> to request the NetBIOS name of the destination host, but could also be caused by a worm called network.vbs. In any case it is classified as a port 137 Scan. It is generated by packets as

those shown by Bryce Alexander in
http://www.sans.org/newlook/resources/IDFAQ/port_137.htm

```
[**] SMB Name Wildcard [**]
05/10-18:08:08.348991 badguy.com:137 -> goodguy.com:137
UDP TTL:119 TOS:0x0 ID:45873
Len: 58
00 DA 00 00 00 01 00 00 00 00 00 00 20 43 4B 41 ............ CKA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 00 00 21 AAAAAAAAAAAAA..!
00 01 ..
```

Defense instructions: disable file sharing on Windows/SAMBA hosts. Since no external IP appear as source hosts, it is evident that port 137 is blocked by a router access list or a perimeter firewall.
Recommendations: enable the signature only for external sources trying to access internal machines, even if this packets should be blocked at the perimeter. This reduces the number of collected records that increases the alert file length and floods the analysts with megabytes of rubbish.

2. SNMP public access Severity: Medium

The SNMP is used to monitor and maintain network devices/hosts. It is a Simple Network Management Protocol based on a clear text password (the so called community name) that distinguishes between a read only and a read/write user.
The snort rules firing these alerts could be the following:

```
alert udp any any -> $HOME_NET 161 (msg:"SNMP public access"; content:"public";
reference:cve,CAN-2002-0012; reference:cve,CAN-2002-0013; sid:1411; rev:2;
classtype:attempted-recon;)
alert tcp any any -> $HOME_NET 161 (msg:"SNMP public access"; flags:A+;
content:"public"; reference:cve,CAN-2002-0012; reference:cve,CAN-2002-0013; sid:1412;
classtype:attempted-recon; rev:4;)
```

Of course, if SNMP from the outside is blocked, only internal hosts trigger this rule.

In Feb 2002 the CERT/CC released an advisory pertaining to a series of vulnerabilities present in various implementations of the protocol that could cause denial of service and/or remote compromise of the systems offering SNMP access.

There are several entries or candidates in the CVE database http://www.cve.mitre.org regarding the SNMP protocol/implementation:

CAN-2002-0012	Vulnerabilities in a large number of SNMP implementations allow remote attackers to cause a denial of service or gain privileges via SNMPv1 trap handling, as demonstrated by the PROTOS c06-SNMPv1 test suite. NOTE: It is highly likely that this candidate will be SPLIT into multiple candidates, one or more for each vendor. This and other SNMP-related candidates will be updated when more accurate information is available.

CAN-2002-0013	Vulnerabilities in the SNMPv1 request handling of a large number of SNMP implementations allow remote attackers to cause a denial of service or gain privileges via (1) GetRequest, (2) GetNextRequest, and (3) SetRequest messages, as demonstrated by the PROTOS c06-SNMPv1 test suite. NOTE: It is highly likely that this candidate will be SPLIT into multiple candidates, one or more for each vendor. This and other SNMP-related candidates will be updated when more accurate information is available.
CAN-2002-0017	Buffer overflow in SNMP daemon (snmpd) on SGI IRIX 6.5 through 6.5.15m allows remote attackers to execute arbitrary code via an SNMP request.
CAN-2002-0053	Buffer overflow in SNMP agent service in Windows 95/98/98SE, Windows NT 4.0, Windows 2000, and Windows XP allows remote attackers to cause a denial of service or execute arbitrary code via a malformed management request. NOTE: this candidate may be split or merged with other candidates. This and other PROTOS-related candidates, especially CAN-2002-0012 and CAN-2002-0013, will be updated when more accurate information is available.
CAN-2002-0069	Memory leak in SNMP in Squid 2.4 STABLE3 and earlier allows remote attackers to cause a denial of service.
CAN-2002-0109	Linksys EtherFast BEFN2PS4, BEFSR41, and BEFSR81 Routers, and possibly other products, allow remote attackers to gain sensitive information and cause a denial of service via an SNMP query for the default community string "public," which causes the router to change its configuration and send SNMP trap information back to the system that initiated the query.
CAN-2002-0302	The Notify daemon for Symantec Enterprise Firewall (SEF) 6.5.x drops large alerts when SNMP is used as the transport, which could prevent some alerts from being sent in the event of an attack.
CAN-2002-0305	Zero One Tech (ZOT) P100s print server does not properly disable the SNMP service or change the default password, which could leave the server open to attack without the administrator's knowledge.
CAN-2002-0478	The default configuration of Foundry Networks EdgeIron 4802F allows remote attackers to modify sensitive information via arbitrary SNMP community strings.
CAN-2002-0540	Nortel CVX 1800 is installed with a default "public" community string, which allows remote attackers to read usernames and passwords and modify the CVX configuration.
CAN-2002-0812	Information leak in Compaq WL310, and the Orinoco Residential Gateway access point it is based on, uses a system identification string as a default SNMP read/write community string, which allows remote attackers to obtain and modify sensitive configuration information by querying for the identification string.
CAN-2002-1048	HP JetDirect printers allow remote attackers to obtain the administrative password for the (1) web and (2) telnet services via an SNMP request to the variable (.iso.3.6.1.4.1.11.2.3.9.4.2.1.3.9.1.1.0.

No external host is involved in these network traces. Top 10 talkers (in the alert file):

No. of detects	Source IP	Hosts probed	Top probed host
13370	130.85.70.177	28	130.85.5.97
9051	130.85.88.181	2	130.85.150.84
5375	130.85.150.198	105	130.85.113.202
4569	130.85.88.203	1	130.85.150.195
4516	130.85.88.159	1	130.85.150.195
4497	130.85.88.207	1	130.85.150.195
4484	130.85.88.145	1	130.85.150.195
4477	130.85.88.136	1	130.85.150.195
2413	130.85.150.245	1	130.85.152.109
729	130.85.88.251	1	130.85.150.195

The accumulated scan file revealed, however, that the host 130.85.5.89 generated as many as 663881 scan records, while SNMP scanning 520 different destinations. This host is not an SNMP management station, of course, because no SNMP manager scans an entire subnet (130.85.200.0/24) starting from 130.85.200.2 up to 130.85.200.222 looking for the SNMP service. The management stations, generally, are configured with the exact device configuration to poll at defined time intervals.
This host, in addition, has been reported as one of the top SMB talkers (hitting 170 hosts) and is the responsible of 44 over 439 ICMP Destination Unreachable (Communication Administratively Prohibited) messages, and hence is suspect to be used by someone who tries to find vulnerable targets.

Recommendations: disable the SNMP service, or, at least, change the default community names (public and private) using more obscure names difficult to guess. Block SNMP from the Internet (it is already done, as no external IP address is a source host). Furthermore, "exposure to SNMP vulnerabilities may be limited by restricting all SNMP access to separate, isolated management networks that are not publicly accessible", as suggested by CERT at http://www.cert.org/advisories/CA-2002-03.html . Mechanisms such as virtual LANs (VLANs) may be used to help segregate traffic on the same physical network.

3. spp_http_decode: IIS Unicode attack detected Severity: High

Code Red (and Code Red II), Nimda and sadmind utilize Unicode encoded characters to try to access the file system, outside the web root directory.
There is no snort rule generating these logs. They are the effect of the activation of the snort pre-processor spp_http_decode.

References: http://online.securityfocus.com/infocus/1232 , http://online.securityfocus.com/bid/1806 , http://www.digitaltrust.it/arachnids/IDS432/event.html

There are 2 entries or candidates in the CVE database http://www.cve.mitre.org regarding IIS Unicode vulnerabilities:

Name	Description
CVE-2000-0884	IIS 4.0 and 5.0 allows remote attackers to read documents outside of the web root, and possibly execute arbitrary commands, via malformed URLs that contain UNICODE encoded characters, aka the "Web Server Folder Traversal" vulnerability.
CAN-2001-0709	Microsoft IIS 4.0 and before, when installed on a FAT partition, allows a remote attacker to obtain source code of ASP files via a URL encoded with Unicode.

The attacks detected by the http_decode snort preprocessor imply that a compromised host is trying to spread the infection. Not all source IP are internal hosts, but several attacks come from internal machines attacking both internal and external addresses. The snort faq at http://www.snort.org/docs/faq.html reveals that " Internal users normal surfing can trigger these alerts in the preprocessor. Netscape in particular has been known to trigger them." So, these could be false positives and could be removed using a BPF filter to ignore outbound http traffic.
Recommendation: better tune the snort sensor, to reduce the false positive events generated by internal users.

A list of all the external offending hosts is given below:

```
Count Source IP
====================
 614 65.92.145.85
 453 217.225.209.131
 440 67.81.229.120
 354 24.101.140.253
 276 80.11.161.54
  26 213.93.221.118
  25 217.167.171.49
  20 130.13.77.117
  16 130.13.166.73
  14 130.13.140.18
  12 61.243.8.61
  12 217.80.93.79
  10 130.13.107.28
   9 202.28.38.209
   8 148.245.230.221
   7 211.90.223.111
   7 130.13.109.53
   5 64.230.87.107
   5 207.230.107.168
   1 194.82.103.37
```

4. ICMP Echo Request L3retriever Ping Severity:Low

This signature is based on the characteristic ping of the L3 Networks security scanner called "Retriever 1.5", however there are reported incidents where legitimate traffic may cause an intrusion detection system to raise "false positive" alerts for this event because this type of ICMP ping seems to be also generated by a Win2K host pinging another

system.
The snort rule generating these logs could be:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP L3retriever Ping"; content:
"ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI"; itype: 8; icode: 0; depth: 32;)
```

Recommendations: disable this signature because it generates only noise.
Correlations: Tod A. Beardsley GCIA practical at
http://www.giac.org/practical/Tod_Beardsley_GCIA.doc

5. MISC Large UDP Packet Severity: Medium

This signature is based on the detection of an UDP packet whose length is greater than 4000 bytes. This could indicate a denial of service attack or the use of a covert channel, or even could be the sign of a multimedia application.
The rule that generated such alerts could be:

```
alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC Large UDP Packet"; dsize: >4000;)
```

According to http://www.digitaltrust.it/arachnids/IDS247/event.html "This event is specific to a vulnerability, but may have been caused by any of several possible exploits. Packet payload is not considered in the signatures used to detect this attack. In most cases this means that the event should be analyzed along with other supporting data before acting on the event."

The source IP addresses generating the alarms are:

```
Count Source IP/(Host name)
=================================================
12708 140.142.8.72 (media-wm-2.cac.Washington.edu)
5810 202.102.249.118 (Zhengzhou Telecom bureau multimedia
information)
3283 140.142.8.71 (media-wn-1.cac.Washington.edu)
1642 211.63.185.15 (Central Data Communication Office)
1107 211.63.185.21 (Central Data Communication Office)
1023 10.16.2.4
 815 208.252.239.125
 716 202.210.163.74
 552 211.233.77.23
 372 205.188.244.227
 125 10.16.2.1
  49 207.189.78.251
  47 202.123.219.153
  27 202.103.102.114
  16 211.233.45.59
   8 216.254.108.19
   1 210.220.161.102
```

While the internal hosts involved are the following:

```
Count Dest IP
=================================================
12708 130.85.153.157
```

```
 5810 130.85.88.140
 3283 130.85.153.159
 2749 130.85.153.179
 1781 130.85.150.209
  716 130.85.152.21
  552 130.85.153.115
  372 130.85.151.108
  182 130.85.153.197
   49 130.85.152.186
   47 130.85.153.187
   27 130.85.153.45
   16 130.85.88.222
    8 130.85.70.133
    1 130.85.153.203
```

All of the 12708 packets coming from 140.142.8.72 are addressed to 130.85.153.157. This is probably the effect of a multimedia application like Netshow, VDOLive or RealPlayer.

The IP addresses in the reserved IANA range (10.x.y.z) are due to crafted packets if they come from the outside and are not filtered on the border, or are the effect of an internal misconfigured host attacking the internal host 130.85.150.209 to various dest ports, including port 0, perhaps trying a denial of service attack against this particular host.

Recommendations: filter audio/video streaming applications on the border router/firewall enabling only connections with known parties.

6. INFO Possible IRC Access Severity:High

The signature that generated these alerts is a homemade one, and I believe a series of false positives could be obtained if it considers only source and/or destination ports without inspecting the packet payload. The rule that fired such alerts could resemble the following:

```
alert tcp $HOME_NET any -> !$HOME_NET 6666:7000 (msg:"INFO Possible IRC Access";
flags:A+;)
```

The brightest star in this alert category is the host 130.85.151.90 that generates 22002 events out of a total of 22079.
This host offers signs of possible compromise: there are possibly processes running on this system that allow unrestricted access to the file system from every host. It hosts possibly an XDCC server, since it generates also 79/80 **`IRC evil - running XDCC`** alerts. An analysis of the XDCC infection is given by Dave Dittrich at http://staff.washington.edu/dittrich/talks/core02/xdcc-analysis.txt.

The IRC ports are very often used for different purposes than for discussing. Several Trojans (Dark FTP, EGO, Maniac rootkit, Moses, ScheduleAgent, SubSeven, Subseven 2.1.4 DefCon 8, The Thing (modified), Trinity, WinSatan Exploit Translation Server, Kazimas, Remote Grab, SubSeven, SubSeven 2.1 Gold) utilize the ports 6667 and 7000 to

distribute "warez" (pirated software) and execute denial of service (DDoS) attacks. In http://www.dslreports.com/faq/4493 you can find: “With IRC in full swing, XDCC bots are common sights in channels these days. An XDCC is a bot that has certain packets uploaded to it. These packets may be anything from the recent game to a good movie. XDCCs are usually r00ted (hacked), and transfer at very high speeds because they are on fast lines”

Recommendations: block IRC/chat at the perimeter of your network.
Utilize tools like fport or lsof to verify which processes utilize the open ports, verify the degree of system compromise and if confirmed remove it from the network and reinstall the OS and the appropriate security patches.

7. INFO MSN IM Chat data Severity:Low

All these alerts are generated by conversations between internal clients and external Messenger servers located on the 64.4.12.0 and 64.4.13.0 networks (DNS domain msgr.hotmail.com).
The following rules could generate such logs:

```
alert tcp [64.4.12.0/24, 64.4.13.0/24] 1863 -> $HOME_NET any (msg:"INFO MSN IM Chat data";
flags:A+;)
alert tcp $HOME_NET any -> [64.4.12.0/24, 64.4.13.0/24] 1863 (msg:"INFO MSN IM Chat data
"; flags:A+;)
```

This should be considered benign traffic and the rule should be dropped from snort config file. The rule triggers when it detects a packet sourced by or sent to port 1863.
There are several entries or candidates in the CVE database http://www.cve.mitre.org regarding MSN IM vulnerabilities:

CAN-2002-0155	Buffer overflow in Microsoft MSN Chat ActiveX Control, as used in MSN Messenger 4.5 and 4.6, and Exchange Instant Messenger 4.5 and 4.6, allows remote attackers to execute arbitrary code via a long ResDLL parameter in the MSNChat OCX.
CAN-2002-0228	Microsoft MSN Messenger allows remote attackers to use Javascript that references an ActiveX object to obtain sensitive information such as display names and web site navigation, and possibly more when the user is connected to certain Microsoft sites (or DNS-spoofed sites).
CAN-2002-0377	Gaim 0.57 stores sensitive information in world-readable and group-writable files in the /tmp directory, which allows local users to access MSN web email accounts of other users who run Gaim by reading authentication information from the files.
CAN-2002-0472	MSN Messenger Service 3.6, and possibly other versions, uses weak authentication when exchanging messages between clients, which allows remote attackers to spoof messages from other users.

Recommendation: Block all the peer-2-peer activity on the border router. However, take into account the following ISS X-Force evaluation:

".NET Messenger has standard port numbers associated with its features, so it is relatively easy to restrict access to some or all of the program.
To prevent just file transfers, disable incoming and outgoing TCP sessions on port 6891.
To prevent Audio/Video conferencing, block the UDP ports 13324 and 13325.
To prevent Application sharing, block the TCP port 1503.
To disable .NET Messenger completely, deny access to hosts in the msgr.hotmail.com subdomain and block TCP port 1863.". The complete document is found at http://documents.iss.net/whitepapers/X-Force_P2P.pdf.

8. AFS - Off-campus activity Severity: Medium

This alert is triggered by packets with destination port 7001 coming from the outside of the University network, like the following rule:

```
alert tcp !$HOME_NET any -> $HOME_NET 7001 (msg:"AFS - Off-campus activity"; flags:A+;)
```

"AFS is a distributed filesystem that enables co-operating hosts (clients and servers) to efficiently share filesystem resources across both local area and wide area networks." A detailed description of the AFS operation is available at : http://www.angelfire.com/hi/plutonic/afs-faq.html .
Port 7000 is used by the file server, while port 7001 is used by the Cache-manager component of the client.

All these alerts correspond to internal clients receiving responses while accessing remote facilities.

The winner in this category is the IP 12.151.57.37 (this address range belongs to ATT Corporation) generating 2565 conversations with only an internal host, the IP 130.85.88.245.

```
# arin 12.151.57.37
AT&T WorldNet Services ATT (NET-12-0-0-0-1)
                                  12.0.0.0 - 12.255.255.255
ATLIGHTSPEED A-LIGHT112-56 (NET-12-151-56-0-1)
                                  12.151.56.0 - 12.151.63.255
```

This host is also triggering 52990 scans records against the same internal host that is probed on almost all UDP ports (17983 events on port 0). This is a clear sign of malicious activity against host 130.85.88.245 and should be reported to AT&T representatives for further analysis and identification and blocking of the real source.

The second most prominent talker is IP 63.215.70.142 that generated 922 conversations with internal IP 130.85.151.95.

```
# arin 63.215.70.142
Level 3 Communications, Inc. LEVEL4-CIDR (NET-63-208-0-0-1)
                                  63.208.0.0 - 63.215.255.255
```

```
Streaming Media Corporation NETBLK-STRM8 (NET-63-215-70-0-1)
                                  63.215.70.0 - 63.215.70.255


# nslookup 63.215.70.142
Server:  dns.my.net.it
Address:  my.net.15.3

Name:    swin13.lax.streamos.com
Address:  63.215.70.142
```

In this second case, perhaps, we have only a false positive: this could be a multimedia applications that utilizes the port used also by AFS.

Another false positive could be the following record, where the source port is completely outside of the range used by AFS systems and reported only because the signature captures all source ports (any).

```
06/13-16:56:44.141610  [**] AFS - Off-campus activity [**] 66.28.225.156:512 ->
130.85.151.95:7001
```

Recommendations: Allow the AFS exchange only with trusted third parties. Block incoming packets with destination port 7001 from all other sites.
References: No CVE entries or candidates relevant to this alert, and no CERT advisories regarding the AFS are reported.

9. INFO Inbound GNUTella Connect request Severity:Low

These are packets coming from remote clients that try to connect to internal Gnutella hosts. Since the destination ports are undefined (any), the signature generating this alert should contain the typical GNUTella content ("GNUTella"), like in the following:

```
alert tcp !$HOME_NET any -> $HOME_NET any (msg:"INFO Inbound GNUTella Connect request ";
flags:A+; content:"GNUTELLA CONNECT"; depth:40;)
```

This package (like Kazaa and Napster) allows peer-to-peer file sharing and is commonly used to share copyrighted material. Additionally, they allow remote, anonymous users access to the file systems of internal machines. Then it raises the possibility that reserved data and files could inadvertently become available to anyone with network connectivity to the systems running this software.

```
#! Destination ports
# grep "Inbound GNUTella" alert | grep request | cut -d \] -f 3 | cut -d "->" -f 2 | cut -
d : -f 2 | sed s/\ //g | sort | uniq -c | sort -rn | more
3902 6346
 214 6406
  18 6391
   3 6348
   1 6405
   1 6364
```

Of the destination hosts receiving these packets, only 2 are really GNUTella servers (they accepted the requests) and I found them using the command:

```
#! Hosts receiving the GNUTella requests
```

```
# grep "Inbound GNUTella" alert | grep request | cut -d \] -f 3 | cut -d "->" -f 2 | cut -
d : -f 1 | sed s/\ //g | sort | uniq -c | sort -rn | more
3329 >130.85.153.142
 469 >130.85.150.209
 225 >130.85.88.240
  46 >130.85.88.215
  41 >130.85.100.157
  14 >130.85.70.149
  12 >130.85.88.178
   3 >130.85.162.198
#! Hosts accepting GNUTella requests
# grep "Inbound GNUTella" alert | grep accept | cut -d \] -f 3 | cut -d "->" -f 1 | cut -d
: -f 1 | sed s/\ //g | sort | uniq -c | sort -rn | more
   7 130.85.150.209
   1 130.85.88.215
```

There is a candidate in the CVE database http://www.cve.mitre.org regarding GNUTella vulnerabilities:

Name	Description
CAN-2001-1004	Cross-site scripting (CSS) vulnerability in gnut Gnutella client before 0.4.27 allows remote attackers to execute arbitrary script on other clients by sharing a file whose name contains the script tags.

The risks associated with this activity have been mapped by ISS in the following diagram taken from the document "Risk Exposure Through Instant Messaging And Peer-To-Peer (P2P) Networks" published by ISS X-Force.

Risks	Unencrypted	File Transfer	Known Buffer Overflows	Remote Control	Known Worms	Spyware	Social Engineering	Misuse of Bandwidth/Copyright Issues	Targeted by known viruses
AIM	X	X	X				X	X	X
.NET	X	X		X	X		X	X	X
Yahoo!	X	X					X	X	
ICQ	X	X	X				X	X	
KaZaA	X	X				X		X	X
gnutella	X	X						X	X

Recommendation: Block all the peer-2-peer activity on the border router. However, take into account the following ISS X-Force evaluations: "Gnutella can use any port for communications on a network, including those that are generally open on a firewall such as port 21, 25, 143, etc. File transfers utilize port 80, therefore even if an administrator were to block Gnutella's default port (6346) both ingoing and outgoing, there is still a method to circumvent security controls. Due to the highly customizable capabilities of Gnutella clients, a network IDS may be the only way to detect users of these clients."

10. ICMP Echo Request Nmap or HPING2

This alert has been generated by a snort signature like the following:

```
alert icmp $HOME_NET any -> any any (msg:"ICMP Echo Request Nmap or HPING2"; dsize: 0;
itype: 8;)
```

The great part of these logs come from the subnet 130.85.152.0/24 (assuming a subnet

schema aligned on a byte boundary) presumably populated by user workstations. Users often try to ping other systems when they perceive a slow network response or just to troubleshoot their application, because they saw the system/network admin using ping during their last intervention.

Additional analysis

Watchlists

After the analysis of the top 10 alerts (in terms of number of generated records) one should consider a priority to analyze the customer watchlists, which have been defined to track down system/networks that are known to host hackers or have a poor security policy in place.

The University defined two of such lists, namely **`Watchlist 000220 IL-ISDNNET-990517`** and **`Watchlist 000222 NET-NCFC.`**
These lists identify two suspicious networks: the 212.179.0.0 and the 159.226.0.0.
These alerts were triggered by:

1. Watchlist 000220 accesses to KaZaa servers on the following internal hosts:

```
Count  Destination IP address:port
============================
2039  130.85.88.162:1214
  35  130.85.150.133:1214
  19  130.85.150.220:1214
   5  130.85.150.113:1214
```

2. accesses to the web server at 130.85.5.108 by 212.179.62.107
3. accesses to web server at 212.179.35.97, 212.179.66.17 and 212.179.35.6 by internal clients
4. conversation between the web server at 159.226.236.10 and the internal host 130.85.153.110.
5. conversation between the web server at 130.85.153.127 and 159.226.49.25
6. accesses from 159.226.236.23 port 8080 to the internal IP 130.85.153.127

A look at the whois database reveals the following information:

```
# ripe 212.179.0.0
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      212.179.0.0 - 212.179.255.255
netname:      IL-ISDNNET-990517
descr:        PROVIDER
country:      IL
admin-c:      NP469-RIPE
tech-c:       TP1233-RIPE
tech-c:       ZV140-RIPE
tech-c:       ES4966-RIPE
status:       ALLOCATED PA
mnt-by:       RIPE-NCC-HM-MNT
changed:      hostmaster@ripe.net 19990517
changed:      hostmaster@ripe.net 20000406
```

```
changed:      hostmaster@ripe.net 20010402
source:       RIPE

route:        212.179.0.0/18
descr:        ISDN Net Ltd.
origin:       AS8551
notify:       hostmaster@bezeqint.net
mnt-by:       AS8551-MNT
changed:      hostmaster@bezeqint.net 20020618
source:       RIPE

person:       Nati Pinko
address:      Bezeq International
address:      40 Hashacham St.
address:      Petach Tikvah  Israel
phone:        +972 3 9257761
e-mail:       hostmaster@isdn.net.il
nic-hdl:      NP469-RIPE
changed:      registrar@ns.il 19990902
source:       RIPE

person:       Tomer Peer
address:      Bezeq International
address:      40 Hashakham St.
address:      Petakh Tiqwah  Israel
phone:        +972 3 9257761
e-mail:       hostmaster@isdn.net.il
nic-hdl:      TP1233-RIPE
changed:      registrar@ns.il 19991113
source:       RIPE

person:       Zehavit Vigder
address:      bezeq-international
address:      40 hashacham
address:      petach tikva 49170 Israel
phone:        +972 52 770145
fax-no:       +972 9 8940763
e-mail:       hostmaster@bezeqint.net
nic-hdl:      ZV140-RIPE
changed:      zehavitv@bezeqint.net 20000528
source:       RIPE

person:       Eran Shchori
address:      BEZEQ INTERNATIONAL
address:      40 Hashacham Street
address:      Petach-Tikva 49170 Israel
phone:        +972 3 9257710
fax-no:       +972 3 9257726
e-mail:       hostmaster@bezeqint.net
nic-hdl:      ES4966-RIPE
changed:      registrar@ns.il 20000309
source:       RIPE
```

This network is held by an Israeli Internet Provider, ISDN NET Ltd of BEZEQ International.

```
# arin 159.226.0.0

OrgName:    The Computer Network Center Chinese Academy of Sciences
OrgID:      CNCCAS

NetRange:   159.226.0.0 - 159.226.255.255
CIDR:       159.226.0.0/16
NetName:    NCFC
NetHandle:  NET-159-226-0-0-1
Parent:     NET-159-0-0-0-0
NetType:    Direct Assignment
```

```
NameServer: NS.CNC.AC.CN
NameServer: GINGKO.ICT.AC.CN
Comment:
RegDate:    1992-06-11
Updated:    1994-07-25

TechHandle: QH3-ARIN
TechName:   Qian, Haulin
TechPhone:  +86 1 2569960
TechEmail:  hlqian@ns.cnc.ac.cn

# ARIN Whois database, last updated 2002-09-05 19:05
# Enter ? for additional hints on searching ARIN's Whois database.
```

This network belongs to the Computer Network Center of the Chinese Academy of Sciences, a well known network to be the source of malicious activities.

Signs of compromised Systems

Among the other, less frequent alerts, the most critical are those indicating the possible presence of a system already compromised. The most prominent are:

```
High port 65535 udp - possible Red Worm - traffic
suspicious host traffic
IRC evil - running XDCC
Possible trojan server activity
High port 65535 tcp - possible Red Worm - traffic
Back Orifice
Port 55850 udp - Possible myserver activity - ref. 010313-1
Port 55850 tcp - Possible myserver activity - ref. 010313-1
Virus - Possible scr Worm
Virus - Possible pif Worm
Virus - Possible MyRomeo Worm
```

The following internal hosts are the sources of various Red Worm traffic (tcp or udp port 65535) and should be carefully analyzed, isolated and eventually sanitized to remove the worm. A tool called adorefind can be used in this case. Adorefind gives you the option to stop the running worm and remove the files from the filesystem.
The "red/adore" launches a program, called "icmp", that listens for a icmp with 77 bytes of data. When a packet of this size is received it forks a root-shell and binds this shell to port 65535.
Adorefind looks for alive hosts of the LOCAL subnet, then tries to connect to port 65535, if the connection is established "red/adore" may be running but can't say if the host is really infected.
If the host's port 65535 is closed, adorefind sends it the MAGIC PACKET (an icmp echo request with 77 bytes of data). If the worm is running, "adore" would fork the shell. Now adorefind will reconnect to port 65535 to check if it is opened or not. The host is "safe" if the port is closed, otherwise we can be sure that the host has been infected.
This tool can be download at :
http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm.
More information on Red Worm is available at: http://www.sans.org/y2k/adore.htm. Red worm infects unix systems, not windows clients; if the host using port 65535 is a

windows hosts, however, the possibility that it has been infected exists. RC1 trojan, aka Remote Control or Rc, is a worm that allows Remote Access. Sins a more recent worm that affects Windows hosts and uses port 65535,too. It uses IRC and ICQ to notify the hacker about its location.
In our case, note that the first 8 hosts accounted for 75% of all the alerts generated.

Red worm Top talkers list:

```
Count Source IP address
======================
 783  130.85.6.49
 627  130.85.6.52
 491  130.85.6.51
 457  130.85.6.50
 327  130.85.6.48
 110  130.85.6.60
 108  130.85.6.53
  61  130.85.6.45
  29  130.85.60.43
  23  130.85.152.21
  22  130.85.152.251
  20  130.85.153.164
   7  130.85.5.96
   6  130.85.149.30
   4  130.85.152.22
   4  130.85.152.179
   4  130.85.149.38
   3  130.85.152.183
   3  130.85.152.174
   3  130.85.152.171
   3  130.85.152.162
   2  130.85.60.151
   2  130.85.153.211
   2  130.85.153.209
   2  130.85.153.203
   2  130.85.153.197
   2  130.85.153.193
   2  130.85.153.189
   2  130.85.153.188
   2  130.85.153.185
   2  130.85.153.180
   2  130.85.153.163
   2  130.85.153.153
   2  130.85.152.249
   2  130.85.152.246
   2  130.85.152.175
   2  130.85.152.169
   2  130.85.152.168
   2  130.85.152.165
   2  130.85.149.56
   2  130.85.149.15
   1  130.85.99.164
   1  130.85.88.245
   1  130.85.6.40
   1  130.85.28.2
   1  130.85.153.216
   1  130.85.153.205
   1  130.85.153.202
   1  130.85.153.195
   1  130.85.153.179
   1  130.85.153.172
   1  130.85.153.170
   1  130.85.153.168
   1  130.85.153.167
   1  130.85.153.162
```

```
   1  130.85.153.161
   1  130.85.153.150
   1  130.85.153.142
   1  130.85.153.141
   1  130.85.153.140
   1  130.85.152.248
   1  130.85.152.216
   1  130.85.152.20
   1  130.85.152.19
   1  130.85.152.186
   1  130.85.152.182
   1  130.85.152.180
   1  130.85.152.178
   1  130.85.152.172
   1  130.85.152.163
   1  130.85.152.160
   1  130.85.152.16
   1  130.85.152.15
   1  130.85.152.13
   1  130.85.151.90
   1  130.85.149.35
   1  130.85.149.18
```

At least a number (16) of the Red Worm alerts are false positives generated by a remote client that randomly choose a tcp ephemeral port of 65535 while accessing the web server at 130.85.5.96 . The conversation is between 130.85.5.96:80 and 204.120.54.1:65535. Host 130.85.151.90 has a server process listening on the port tcp/65535 that accepted a connection by 130.85.28.2, so perhaps it is infected by Red Worm.

The Trojan server activity generated inside the University network can be attributed to the following hosts source hosts:

```
#grep trojan alert|cut -d "]" -f3 |cut -d ">" -f1 |cut -d \: -f1|sort|uniq -c | sort -rn
Count Source IP address
======================
  23  130.85.5.83
  13  130.85.5.88
   5  130.85.70.177
   5  130.85.5.42
   5  130.85.5.19
   4  130.85.88.162
   1  130.85.88.245
   1  130.85.28.2
   1  130.85.253.10
   1  130.85.151.90
```

The following hosts show sign of compromise by at least the SubSeven Trojan, aka Sub7 or Backdoor_G: 130.85.5.19, 130.85.5.88, 130.85.70.177, 130.85.5.42, 130.85.151.90 and 130.85.88.245.

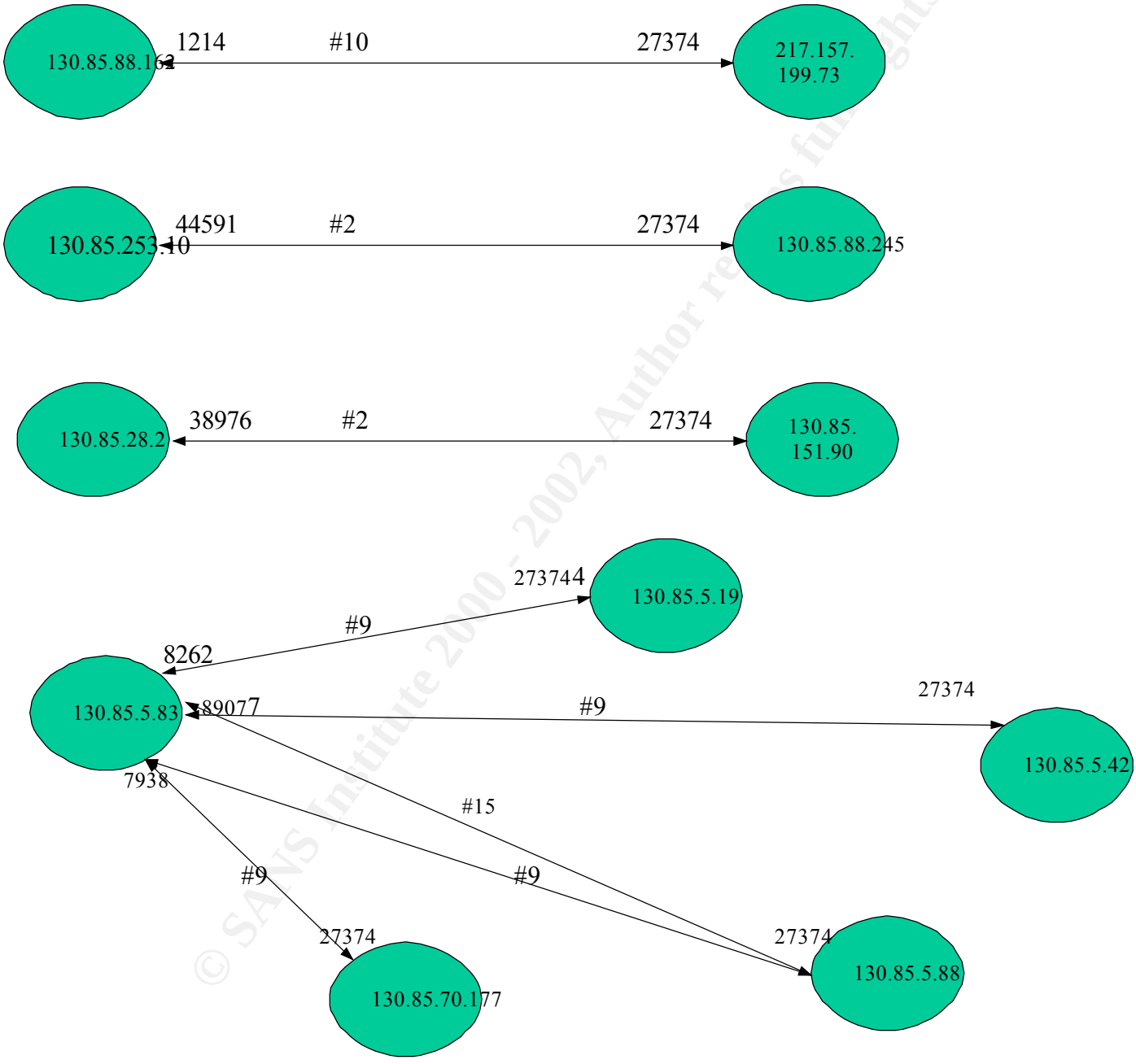
The host 130.85.28.2 engaged a myserver tcp conversation with 130.85.151.90, that has a process listening on tcp port 55850: this host should be carefully analyzed and the presence of the Trojan should be confirmed using tools like fport or netstat. In such a case the system should be sanitized appropriately.

The data analyzed revealed the presence of three known Viruses on the e-mails stored on the two POP2 servers 130.85.6.7 and 130.85.6.39: MyRomeo, pif and scr viruses. I

recommend the security team of the University to utilize a centrally managed anti-virus solution. This anti-virus should have the capability of automatic signature update from the Internet, that should be used by a certain number of internal systems that are in charge to distribute the new signatures to all the client hosts at least every 8 hours.

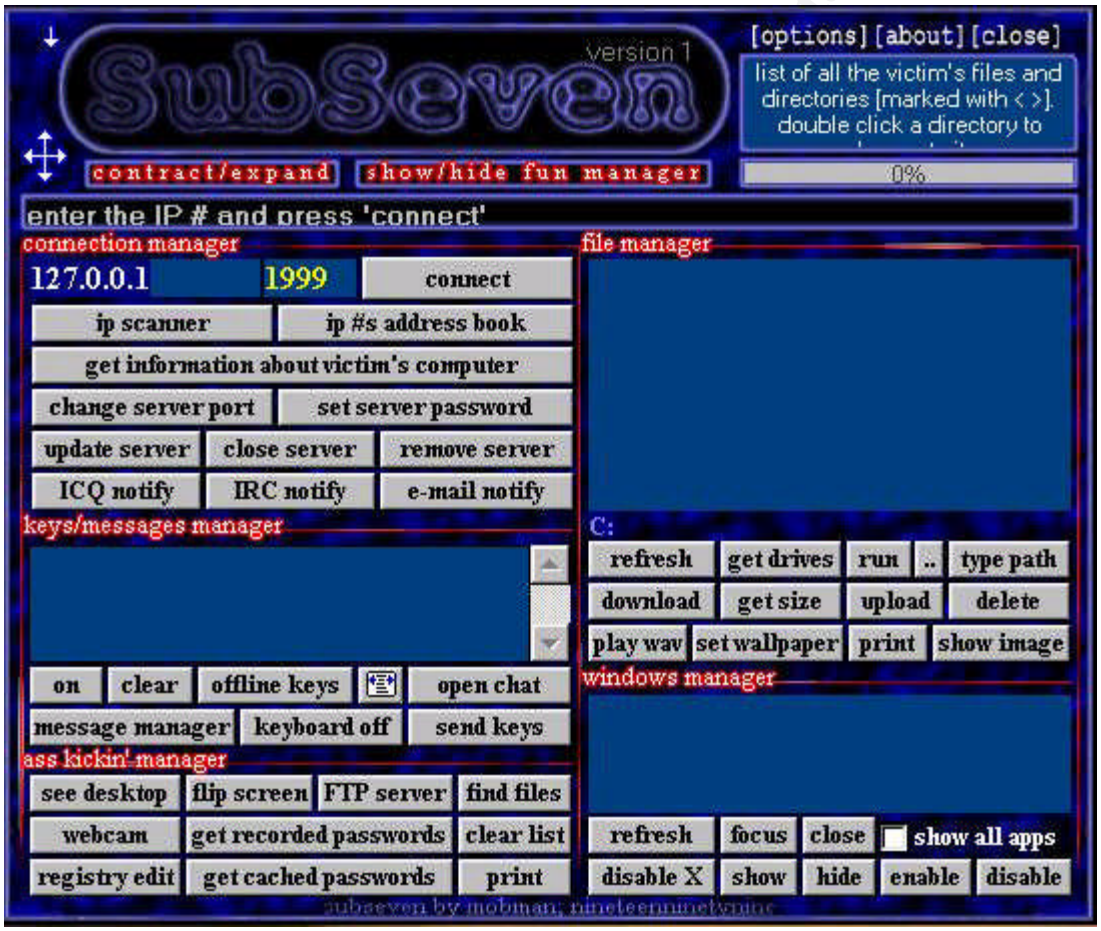
Host 130.85.157.250 generated all the NIMDA alerts but one that is attributed to host 130.85.151.90. These hosts are compromised by the NIMDA worm and try to spread the infection. They should be disconnected from the network, the presence of the worm confirmed and appropriate actions should be taken to remove the NIMDA code, eventually reinstalling a fresh copy of the operating system with all the relevant security patches produced by the manufacturer.

Link analysis of the Trojan server activity



In this figure, each host is represented by a circle containing the host IP address. The conversations between two hosts are represented by the double arrowed lines connecting

two circles, while the number at the center of the line indicates how many alert record correspond to this conversation. The numbers located at each extreme of the line are the port numbers used for the conversation.
From the previous image, the following facts emerge:
host 217.157.199.73 uses an ephemeral port 27374 to access the KaZaa server at 130.85.88.162 and snort fires a false alarm
host 130.85.253.10 connects to the SubSeven server at 130.85.88.245
host 130.85.28.2 connects to the Subseven server at 130.85.151.90
host 130.85.5.83 connects to various internal hosts infected by Subseven. Using an interface such as that shown in the following picture, this user can perform several management actions on the infected hosts, including full remote host control (For more info on SubSeven see at: http://www.sans.org/newlook/resources/IDFAQ/subseven.htm).



Scan files Analysis

SYN-FIN scans:

There are only two records reporting this type of scan :

```
Jun 13 18:17:00 64.4.124.151:3193 -> 130.85.88.165:1269 SYNFIN 1*****SF RESERVEDBITS
Jun 15 17:32:23 65.69.223.128:7000 -> 130.85.153.178:7001 SYNFIN 12****SF RESERVEDBITS
```

VECNA Scans:

This type of scan is generated by TCP packets with the following flags:

1. URG bit alone
2. PUSH bit alone
3. FIN+URG bits
4. PUSH+FIN bits
5. URG+PUSH bits

These type of scans are intercepted by the spp_stream4.c module of snort, or, as is our case, by the spp_portscan.c since the stream4 is not enabled on this sensor.

In http://marc.theaimsgroup.com/?l=snort-users&m=96449963319115&w=2 this kind of scans (along with FULL XMAS scans) has been reported as generated by systems in the daemon.co.uk domain. Daemon.net people were asked about this and they responded stating they have had a problem with a device mangling packets.

None of the University traces have been originated by machines inside the Daemon Net domains. The following hosts show this type of activity (along with the record count):

```
grep VECNA scans | cut -d ">" -f1 |cut -d " " -f 4|cut -d \: -f1|sort |uniq -c|sort -rn
  19 65.69.223.128
   6 148.63.236.115
   4 64.4.124.151
   2 24.112.58.210
   2 148.63.222.102
   1 148.64.8.182
   1 148.63.145.79
```

XMAS Scans:

Only 12 records exhibited the typical Christmas scan pattern:

```
Jun 12 21:18:31 24.112.58.210:2656 -> 130.85.150.209:6346 FULLXMAS 1*UAPRSF RESERVEDBITS
Jun 12 21:21:56 24.112.58.210:2656 -> 130.85.150.209:6346 FULLXMAS *2UAPRSF RESERVEDBITS
Jun 13 10:10:29 130.85.28.2:38989 -> 130.85.151.90:1 XMAS **U*P**F
Jun 14 03:44:31 192.168.0.100:0 -> 130.85.150.133:0 FULLXMAS 12UAPRSF RESERVEDBITS
Jun 15 17:30:15 65.69.223.128:7000 -> 130.85.153.178:7001 XMAS 1*U*P**F RESERVEDBITS
Jun 15 17:32:00 65.69.223.128:30984 -> 130.85.153.178:37816 FULLXMAS 12UAPRSF RESERVEDBITS
Jun 15 17:37:53 65.69.223.128:0 -> 130.85.153.178:0 FULLXMAS 12UAPRSF RESERVEDBITS
Jun 15 17:43:49 65.69.223.128:28530 -> 130.85.153.178:11893 XMAS *2U*P**F RESERVEDBITS
Jun 15 17:51:59 65.69.223.128:7000 -> 130.85.153.178:7001 XMAS 1*U*P**F RESERVEDBITS
Jun 15 17:52:09 65.69.223.128:7000 -> 130.85.153.178:7001 FULLXMAS **UAPRSF
Jun 15 17:52:23 65.69.223.128:7001 -> 130.85.153.178:7000 XMAS 1*U*P**F RESERVEDBITS
Jun 16 11:35:53 12.224.236.145:65329 -> 130.85.88.162:1214 XMAS 12U*P**F RESERVEDBITS
```

NULL Scans:

Many more records revealed a null scan: there are 216 such scans. The most active source of these scans is host 65.69.223.128 that generated 179 events (most of them from port 0). This host, therefore, is actively scanning the University networks with a variety of techniques (SF, XMAS, NULL scans).
A look at the dns and whois databases reveals its origin: it is an adsl user connected via

Southwestern Bell.

```
# arin 65.69.223.128
Southwestern Bell Internet Services SBIS-5BLK (NET-65-64-0-0-1)
                                  65.64.0.0 - 65.71.255.255
PPPoX Pool - HSTNTXRBACK11 SBCIS-10161-144845 (NET-65-69-220-0-1)
                                  65.69.220.0 - 65.69.223.255

# ARIN Whois database, last updated 2002-09-11 19:05
# Enter ? for additional hints on searching ARIN's Whois database.

# nslookup 65.69.223.128
Server:  dns.my.net.it
Address:  x.y.15.3

Name:    adsl-65-69-223-128.dsl.hstntx.swbell.net
Address:  65.69.223.128
```

The top 10 lists generated analyzing the accumulated scan files are reported here along with the number of events recorded. These lists contain the top source IP, the top destination IP and the top scanned ports, with the accumulated record count.

```
==> scans.sourceip <==
667972 130.85.5.89
407697 130.85.60.43
52990 12.151.57.37
52547 130.85.6.49
49209 130.85.6.45
41365 130.85.28.2
41103 130.85.6.52
37698 130.85.253.10
33228 130.85.6.50
31912 130.85.6.51


==> scans.destip <==
90701 130.85.88.245
61416 130.85.1.3
41411 130.85.151.90
33476 130.85.1.4
31916 130.85.11.7
26356 130.85.11.6
26062 130.85.6.45
21768 130.85.60.43
17757 130.85.5.55
17358 130.85.5.50

==> scans.destport <==
666030 161
219913 80
150396 7001
97175 53
93523 7000
59463 137
58032 0
49851 6970
36877 1214
24281 7003
```

OOS files analysis

The Out of spec files report invalid packets dumps. These packets have been qualified for

the presence of the following invalid signatures:

1. TCP packets with an invalid combination of flags in the TCP header
2. Presence of Congestion management flags
3. Initial fragment (frag offset 0x0) and DF and MF flags set and fragment size invalid (0x22=34 bytes, not a multiple of 8 bytes)

In addition other signs of packet corruption/crafting are evident:

1. TCP packets with destination port 0
2. TCP packet with source port 0
3. TCP packets with a high number of Opt 21 or Opt 53 in the TCP header

The following hosts generated these packets :

```
grep '\-\>' oos | cut -d " " -f2| cut -d \: -f1|sort |uniq -c |sort -rn
Count Source IP
==================
   7 64.4.124.151
   6 24.112.58.210
   5 65.65.224.233
   4 195.101.94.208
   2 24.120.177.22
   2 193.6.40.86
   1 68.80.114.202
   1 68.50.107.141
   1 66.25.185.163
   1 65.42.230.217
   1 62.99.143.179
   1 62.99.143.178
   1 62.78.169.87
   1 12.224.236.145
   1 12.217.65.38
```

In the following paragraph an analysis of the top 5 oos talkers is reported.

Host 64.4.124.151

A look at the DNS and whois databases reveals the following information:

```
#arin 64.4.124.151
Ntelos North Cisco DSL DHCP Range #2 CFW-64-4-124-NNC (NET-64-4-124-0-1)
                                64.4.124.0 - 64.4.124.255
CFW Communications CFW-BLK-2 (NET-64-4-96-0-1)
                                64.4.96.0 - 64.4.127.255

# ARIN Whois database, last updated 2002-09-11 19:05
# Enter ? for additional hints on searching ARIN's Whois database.
# nslookup 64.4.124.151
Server:  dns.ccsem.infn.it
Address:  my.net.15.3

Name:    64-4-124-151.dmt.ntelos.net
Address:  64.4.124.151
```

This host generated the following logs:

```
06/13-17:39:58.851407 64.4.124.151:3193 -> MY.NET.88.165:1269
06/13-17:46:22.699466 64.4.124.151:0 -> MY.NET.88.165:3193
06/13-17:54:54.956901 64.4.124.151:4 -> MY.NET.88.165:3193
06/13-18:00:01.789438 64.4.124.151:3193 -> MY.NET.88.165:1269
06/13-18:16:02.185414 64.4.124.151:3193 -> MY.NET.88.165:1269
06/13-18:18:59.662527 64.4.124.151:3193 -> MY.NET.88.165:1269
06/13-18:36:41.669086 64.4.124.151:3193 -> MY.NET.88.165:1269
```

Also the scans and the alert files report the activity of this host with respect to host my.net.88.165 on both port 1269 (all packets came from port 3139) and 3193 (from port 0 and 4). The different timestamps between the oos file and the scans data may be indicative that these files were generated by different systems.
Port 1269 is a know Trojan port, see the details at
http://www.simovits.com/trojans/tr_data/y1024.html.
So this could be the indication that the oos host was looking for a Matrix compromised host.
Interestingly enough, there is some sequence in the ack number of the packets generated by this host, and all the packets but one have both CWR and ECN-echo bits set:

```
21**R**U Seq: 0xBCCA1D8   Ack: 0x7D86   Win: 0x5010
21**RP*U Seq: 0x4F50D80   Ack: 0x1D87D87   Win: 0x5010
21**R*** Seq: 0x4F50FC7   Ack: 0x31D87D88   Win: 0x5010
21**R**U Seq: 0x11122   Ack: 0xD1D87D89   Win: 0x5010
21**RP*U Seq: 0x1566B78C   Ack: 0x7D8C   Win: 0x5010
*1SF**** Seq: 0x163141D8   Ack: 0x7D8D7EAC   Win: 0x5010
21**RP*U Seq: 0x1ADFD1D8   Ack: 0x927D90   Win: 0x5010
```

Host 24.112.58.210

A look at the DNS and whois databases reveals the following information:
This host is not registered in the dns database, but the ARIN whois server gives some useful information.

```
# arin 24.112.58.210

OrgName:    Rogers Cable Inc.
OrgID:      ROCB

NetRange:   24.112.0.0 - 24.112.255.255
CIDR:       24.112.0.0/16
NetName:    ROGERS-CAB-1
NetHandle:  NET-24-112-0-0-1
Parent:     NET-24-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.WLFDLE.RNC.NET.CABLE.ROGERS.COM
NameServer: NS2.WLFDLE.RNC.NET.CABLE.ROGERS.COM
NameServer: NS1.YM.RNC.NET.CABLE.ROGERS.COM
NameServer: NS2.YM.RNC.NET.CABLE.ROGERS.COM
Comment:
RegDate:
Updated:    2002-08-20

TechHandle: AD30-ARIN
TechName:   Budd, Paul
TechPhone:  +1-416-935-4729
TechEmail:  abuse@rogers.com

OrgTechHandle: IPMAN-ARIN
OrgTechName:   IP Management
OrgTechPhone:  +1-416-935-7291
OrgTechEmail:  ipmanage@rogers.wave.ca
```

Here are the records generated by this host:

```
06/12-21:18:33.952340 24.112.58.210:166 -> MY.NET.150.209:2656
06/12-21:20:30.868375 24.112.58.210:2656 -> MY.NET.150.209:6346
06/12-21:23:55.685430 24.112.58.210:2656 -> MY.NET.150.209:6346
06/12-21:24:24.400833 24.112.58.210:2656 -> MY.NET.150.209:6346
06/12-21:25:35.910653 24.112.58.210:2656 -> MY.NET.150.209:6346
06/12-21:32:43.115158 24.112.58.210:2656 -> MY.NET.150.209:6346
```

His activity started June 12nd at 21:18:33 and terminated 14 minutes later. Similar to top oos #1, the second most active talker produces packets with invalig TCP flag combinations, and (with the exception of the first packet) always raises one or both the Congestion bits. Similar activity is reported in the scans file (20 records with this source IP), but the timestamps are different.

```
**SF***U Seq: 0x18CA0546   Ack: 0xA9D0ADF   Win: 0x5018
*1SFRPAU Seq: 0x690547   Ack: 0xF2A70AEA   Win: 0x5018
2*SFRPAU Seq: 0x54D2CF3   Ack: 0x6B0AF3   Win: 0x5018
21SF***U Seq: 0xA6054D   Ack: 0xD6210AF4   Win: 0x5018
*1SF**AU Seq: 0x54F   Ack: 0x294B0AF5   Win: 0x5018
21*FRP*U Seq: 0x557DF8E   Ack: 0xAFF9BB5   Win: 0x5018
```

The destination ports of this attack are 2656 (registered to Kana service) and 6346. The last is the most interesting, since it is an indication that a Gnutella server is looked for. This is the case for the destination host, because the alert file gives the following traces that indicate the presence of a responding GNUTella server on system at IP 130.85.150.209:

```
06/13-19:10:25.203850  [**] INFO Inbound GNUTella Connect accept [**] 130.85.150.209:6346 -
> 65.92.188.60:1689
06/13-19:25:31.625179  [**] INFO Inbound GNUTella Connect accept [**] 130.85.150.209:6346 -
> 24.200.129.182:1792
06/13-19:33:38.293698  [**] INFO Inbound GNUTella Connect accept [**] 130.85.150.209:6346 -
> 217.0.7.42:3682
06/13-21:54:42.473932  [**] INFO Inbound GNUTella Connect accept [**] 130.85.150.209:6346 -
> 80.195.151.199:4166
06/14-14:35:18.871945  [**] INFO Inbound GNUTella Connect accept [**] 130.85.150.209:6346 -
> 24.229.204.212:1197
06/14-17:36:50.871131  [**] INFO Inbound GNUTella Connect accept [**] 130.85.150.209:6346 -
> 63.67.241.32:1564
06/14-17:36:56.879972  [**] INFO Inbound GNUTella Connect accept [**] 130.85.150.209:6346 -
> 63.67.241.32:1564
```

Host 65.65.224.233

A look at the DNS and whois databases reveals the following information:

```
# nslookup 65.65.224.233
Server:  dns.x.y.it
Address:  x.y.15.3

Name:    adsl-65-65-224-233.dsl.spfdmo.swbell.net
Address:  65.65.224.233

# arin 65.65.224.233
Southwestern Bell Internet Services SBIS-5BLK (NET-65-64-0-0-1)
                                  65.64.0.0 - 65.71.255.255
rback1 SBCIS-100125-133432 (NET-65-65-224-0-1)
                                  65.65.224.0 - 65.65.225.255
```

This host generated all the oos packets in a couple of minutes:

```
06/10-16:59:19.796245 65.65.224.233 -> MY.NET.88.162
06/10-16:59:22.453703 65.65.224.233 -> MY.NET.88.162
06/10-16:59:28.630248 65.65.224.233 -> MY.NET.88.162
06/10-16:59:40.914744 65.65.224.233 -> MY.NET.88.162
06/10-17:00:52.571538 65.65.224.233 -> MY.NET.88.162
```

These traces mean retransmission: the same packet is sent once at 06/10-16:59:19, then 3, 6, 12 and 12 seconds later. These packets show the DF+MF invalid bit combination, they are the first fragment of a packet with invalid frag size (0x22).
The alert and scans files don't contain any record regarding this host.

Host 195.101.94.208

A look at the DNS and whois databases reveals the following information:

```
# ripe 195.101.94.208
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      195.101.94.0 - 195.101.94.255
netname:      FR-ECHO
descr:        Socite ECHO
country:      FR
admin-c:      CR308-RIPE
tech-c:       CR308-RIPE
status:       ASSIGNED PA
notify:       addr-reg@rain.fr
mnt-by:       RAIN-TRANSPAC
changed:      noc@rain.fr 19970514
source:       RIPE

route:        195.101.94.0/24
descr:        ECHO
origin:       AS8891
mnt-by:       OLEANE-NOC
changed:      hostmaster@oleane.net 19980929
source:       RIPE

person:       Christophe RUELLE
address:      ECHO
address:      20 Parc des hautes Technologies
address:      06250 MOUGINS
phone:        +33 4 92 28 32 00
fax-no:       +33 4 92 28 32 01
e-mail:       ruelle@echo.fr
nic-hdl:      CR308-RIPE
notify:       addr-reg@rain.fr
mnt-by:       RAIN-TRANSPAC
changed:      nathalie@rain.fr 20000324
source:       RIPE


GWCCSEM # nslookup 195.101.94.208
Server:  dns.my.net.it
Address:  my.net.15.3

Name:    x1crawler1-1-0.x-echo.com
Address:  195.101.94.208
```

As the name indicates, this looks like a crawler of a search engine. Grepping a little on the oos, scan and alert files, one can see that all the destination ports of these packets are port 80 (indicative of a web server).

All the packets contain CWR and ECN-echo bits set plus the initial SYN bit: this can be

interpreted as a Queso scan because this tool uses such a pattern as one od its OS fingerprinting methodologies. More probably, the crawler is trying to contact some web server to index its pages. RFC 3168, "The Addition of Explicit Congestion Notification (ECN) to IP", states that "For a SYN packet, the setting of both ECE and CWR in the ECN-setup SYN packet is defined as an indication that the sending TCP is ECN-Capable". For a complete description of the ECN mechanism see at: ftp://ftp.isi.edu/in-notes/rfc3168.txt .

```
# grep 195.101.94.208 oos
06/12-17:17:49.919492 195.101.94.208:2102 -> MY.NET.5.95:80
06/14-03:26:59.191077 195.101.94.208:2033 -> MY.NET.5.95:80
06/14-07:28:33.002101 195.101.94.208:1107 -> MY.NET.150.83:80
06/14-07:43:16.003243 195.101.94.208:1385 -> MY.NET.5.96:80
# grep 195.101.94.208 scans
Jun 12 17:15:50 195.101.94.208:2102 -> 130.85.5.95:80 SYN 12****S* RESERVEDBITS
Jun 14 03:25:00 195.101.94.208:2033 -> 130.85.5.95:80 SYN 12****S* RESERVEDBITS
Jun 14 07:26:33 195.101.94.208:1107 -> 130.85.150.83:80 SYN 12****S* RESERVEDBITS
Jun 14 07:41:16 195.101.94.208:1385 -> 130.85.5.96:80 SYN 12****S* RESERVEDBITS
# grep 195.101.94.208 alert|grep -v spp
06/12-17:15:50.893910  [**] Queso fingerprint [**] 195.101.94.208:2102 -> 130.85.5.95:80
06/14-03:25:00.039621  [**] Queso fingerprint [**] 195.101.94.208:2033 -> 130.85.5.95:80
06/14-07:26:33.832287  [**] Queso fingerprint [**] 195.101.94.208:1107 ->
130.85.150.83:80)
06/14-07:41:16.832513  [**] Queso fingerprint [**] 195.101.94.208:1385 -> 130.85.5.96:80
```

Host 24.120.177.22

A look at the DNS and whois databases reveals the following information:

```
# nslookup 24.120.177.22
Server:  dns.my.net.it
Address:  my.net.15.3

Name:    cm022.177.120.24.lvcm.com
Address:  24.120.177.22

# arin 24.120.177.22

OrgName:    Cox Communications Inc.
OrgID:      CXA

NetRange:   24.120.0.0 - 24.120.255.255
CIDR:       24.120.0.0/16
NetName:    COX-ATLANTA-6
NetHandle:  NET-24-120-0-0-1
Parent:     NET-24-0-0-0-0
NetType:    Direct Allocation
NameServer: PRIME-BE1.LVCABLEMODEM.COM
NameServer: NEWS.LVCABLEMODEM.COM
Comment:
RegDate:
Updated:    2002-08-21

TechHandle: IC146-ARIN
TechName:   Cox Communications, Inc
TechPhone:  +1-404-269-7626
TechEmail:  abuse@cox.net

OrgAbuseHandle: IC146-ARIN
OrgAbuseName:   Cox Communications, Inc
OrgAbusePhone:  +1-404-269-7626
OrgAbuseEmail:  abuse@cox.net
```

```
OrgTechHandle: WILLI-ARIN
OrgTechName:   Williams, Matt
OrgTechPhone:  +1-404-843-5827
OrgTechEmail:  matt.williams@cox.com
```

There are only 2 packets in the oos file that were generated by this host. There is no relevant record in the scans or alert file generated by this host.

```
06/10-01:59:45.549380 24.120.177.22:4130 -> MY.NET.88.162:1045
TCP TTL:111 TOS:0x0 ID:21057  DF
21SF**AU Seq: 0x22918B   Ack: 0x100059A3   Win: 0x2037
10 22 04 15 00 22 91 8B 10 00 59 A3 01 F3 20 37  ."...".....Y... 7
02 00 1D F6 13 01 74 00 FD FE 00 00              ......t.....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/10-02:01:18.449230 24.120.177.22:4130 -> MY.NET.88.162:1045
TCP TTL:111 TOS:0x0 ID:34881  DF
21SF**AU Seq: 0x2223B5   Ack: 0x10005D1E   Win: 0x2037
10 22 04 15 00 22 23 B5 10 00 5D 1E 01 F3 20 37  ."..."#...]... 7
02 00 6E 00 A5 00 24 FE 11 FF 00 00 00 00 E3 08  ..n...$.........
```

Summary and suggestions

The sensor currently used at the University is producing more than 100,000 alerts and 400,000 scan records on average per day. It is infeasible to examine all of these alerts; hence, a better tuning of the system is required. The current version of Snort provides a way to prioritize alerts. The analysts should focus on the high priority alerts and not be disturbed by the high volume of lower priority events.

For example, use portscan-ignorehosts to ignore TCP SYN and UDP "scans" from specific networks or hosts to reduce false alerts. In particular, exclude those hosts that are know to generate false alerts like DNS and NNTP servers.

A number of dangerous services are used and provided at the University. These services include the peer-2-peer file sharing services such as Gnutella and Kazaa, which are notorious for being used to illegally exchange copyrighted material. These services should be blocked at the external perimeter by a router access list and/or a stateful firewall.

Investigate more closely all the hosts that have shown signs of system compromise and that possibly host a Trojan like SubSeven, myserver, Red Worm or NIMDA. Disconnect these hosts from the network, perform a thorough analysis, clean the infected files and apply all the relevant security patches to them, disabling all the unneeded services that a default installation produces.

Use an anti-virus solution both on the user population PC's and on the mail server. The most popular anti-viruses are continuously updated: configure all your host for periodic automatic updates (at least once a day).

Educate your users to be informed and to err on the side of security: they should inform the University Security Team every time they note something strange is happening in their systems or user accounts. They should not be prone to share files and folders with extraneous, they should not share passwords or email them to coworkers. They should protect their systems with personal firewall and anti-viruses and avoid to use peer-2-peer software.

Appendices: Data Analysis Procedures

The data files were concatenated using the following standard unix command procedures:

```
for i in *.gz; do
       gzip –d $i
done
for i in alert*; do
       cat $i >>alert
done
for i in oos*; do
       cat $i >>oos
done
for i in scans*; do
       cat $i >>scans
done
```

Then I extracted the ordered list of the most involved destination host, destination port and source host with the following Unix script:

```
#!/bin/ksh
# It is necessary to collect a tally of source host and destination IPs and ports.
# With this data, the Top 10 lists can be formed.
# This procedure uses only standard unix tools.
echo "Give the input filename > "
read file
#
grep "\[\*\*\]" $file | grep -v spp_portscan | cut -d \> -f 2 \
       | cut -d : -f 1 | sed s/\ //g | sort | uniq -c | sort -nr > tallies.dstips.log

grep "\[\*\*\]" $file | grep -v spp_ | cut -d \> -f 2 | cut -d : -f 2 -s \
       | sed s/\ //g | sort | uniq -c | sort -nr > tallies.dstports.log

grep "\[\*\*\]" $file | grep -v spp_ |    cut -d \] -f 3 | cut -d "->" -f 1 \
       | cut -d : -f 1 \
       | sed s/\ //g > tallies.srcips.log.unsorted
grep End $file | cut -d \] -f 2 | cut -d : -f 2 \
       | cut -d " " -f 6 >> tallies.srcips.log.unsorted
grep spp_stream4 $file |cut -d } -f2 | cut -d "->" -f1 | cut -d : -f 1\
       | sed s/\ //g >> tallies.srcips.log.unsorted
#
cat tallies.srcips.log.unsorted | sort | uniq -c | sort -nr > tallies.srcips.log
rm tallies.srcips.log.unsorted
```

Here I consider each portscan an event detected by the “End of portscan” statement in the snort alert file. This way I avoid to cont a single event multiple times (each portscan produces several alert lines: portscan detected, portscan status, End of portscan).

In the alert files files analyzed there are not spp_stream4 output lines: perhaps the stream4

processor is not used at the University, or such lines have been removed by someone prior to posting the data on the web.

```
#!/bin/sh
# The tally of each alert type is needed.
# This procedure accomplishes such a task with standard unix tools.
echo "Give the input filename > "
read file

grep "\[\*\*\]" $file | grep -v spp_portscan | cut -d \] -f 2,3 \
       | tr -d "**" | cut -d \[ -f 1 | sort | uniq -c|sort -rn >tally-alert-type.log
grep End $file | cut -d \] -f 2 | cut -d : -f 2,3 >> tally-alert-type.log

grep "\[\*\*\]" $file | grep -v spp_portscan | cut -d \] -f 2,3 \
       | tr -d "**" | cut -d \[ -f 1 | sort  -u >tally-alert-type.txt
```

Looking at the alert file, it is clear that there are differences with the standard snort rule base. I then downloaded it from http://www.snort.org/dl/signatures/snortrules-stable.tar.gz and compared the collected alert types with the official distribution. It was possible then to find the customization made by the security personnel at the University.

The procedure I wrote to compare the two lists scans the alerts reported and verifies if the message text is the same in the official distribution. If it doesn't find the same message text, prints the message followed by a colon and a zero, otherwise it prints the number of times it found the same text in the official snort signatures. The check procedure is the following:

```
#!/bin/sh
for i in *.rules; do
       cat $i >> all.rules
done
cat all.rules |grep alert| cut -d \" -f2 > alert.txt
while read i
       do
       echo  -n $i ":"
       grep """$i""" alert.txt |wc -l
done < tally-alert-type.txt
```

The result is the following:

```
# ./check
AFS - Off-campus activity :         0
Attempted Sun RPC high port access :         0
Back Orifice :         0
EXPLOIT NTPDX buffer overflow :         0
EXPLOIT x86 NOOP :         0
EXPLOIT x86 setgid 0 :         0
EXPLOIT x86 setuid 0 :         0
EXPLOIT x86 stealth noop :         0
FTP DoS ftpd globbing :         0
High port 65535 tcp - possible Red Worm - traffic :         0
High port 65535 udp - possible Red Worm - traffic :         0
ICMP Destination Unreachable (Communication Administratively Prohibited) :         1
ICMP Destination Unreachable (Host Unreachable) :         1
ICMP Destination Unreachable (Protocol Unreachable) :         1
ICMP Echo Request BSDtype :         0
ICMP Echo Request CyberKit 2.2 Windows :         0
ICMP Echo Request Delphi-Piette Windows :         0
ICMP Echo Request L3retriever Ping :         0
ICMP Echo Request Nmap or HPING2 :         0
```

```
ICMP Echo Request Windows :         0
ICMP Fragment Reassembly Time Exceeded :         1
ICMP Router Selection :         1
ICMP traceroute :         2
IDS552/web-iis_IIS ISAPI Overflow ida nosize :         0
INFO - Possible Squid Scan :         0
INFO FTP anonymous FTP :         0
INFO Inbound GNUTella Connect accept :         0
INFO Inbound GNUTella Connect request :         0
INFO MSN IM Chat data :         0
INFO Napster Client Data :         0
INFO Outbound GNUTella Connect accept :         0
INFO Outbound GNUTella Connect request :         0
INFO Possible IRC Access :         0
IRC evil - running XDCC :         0
Incomplete Packet Fragments Discarded :         0
MISC Large UDP Packet :         1
MISC PCAnywhere Startup :         0
MISC source port 53 to <1024 :         1
MISC traceroute :         0
NIMDA - Attempt to execute cmd from campus host :         0
NMAP TCP ping! :         0
Null scan! :         0
Port 55850 tcp - Possible myserver activity - ref. 010313-1 :         0
Port 55850 udp - Possible myserver activity - ref. 010313-1 :         0
Possible trojan server activity :         0
Probable NMAP fingerprint attempt :         0
Queso fingerprint :         0
SCAN FIN :         1
SCAN Proxy attempt :         0
SCAN Synscan Portscan ID 19104 :         0
SCAN XMAS :         1
SMB Name Wildcard :         0
SNMP public access :         2
SUNRPC highport access! :         0
TFTP - External UDP connection to internal tftp server :         0
TFTP - Internal UDP connection to external tftp server :         0
UDP SRC and DST outside network :         0
Virus - Possible MyRomeo Worm :         7
Virus - Possible pif Worm :         1
Virus - Possible scr Worm :         1
WEB-CGI formmail access :         1
WEB-CGI redirect access :         1
WEB-CGI scriptalias access :         1
WEB-FRONTPAGE _vti_rpc access :         1
WEB-IIS Unauthorized IP Access Attempt :         1
WEB-IIS Unicode2.pl script (File permission canonicalization :         1
WEB-IIS _vti_inf access :         1
WEB-IIS view source via translate header :         1
WEB-MISC 403 Forbidden :         0
WEB-MISC Attempt to execute cmd :         0
WEB-MISC compaq nsight directory traversal :         0
WEB-MISC http directory traversal :         2
Watchlist 000220 IL-ISDNNET-990517 :         0
Watchlist 000222 NET-NCFC :         0
X11 outgoing :         0
spp_http_decode: CGI Null Byte attack detected :         0
spp_http_decode: IIS Unicode attack detected :         0
suspicious host traffic :         0
```