GIAC Certified Intrusion Analyst (GCIA) Practical Assignment v3.4

Vilaiporn Taweelappontong

Contents

Paper Abstract

This paper is prepared to fulfil GCIA certification requirement. The paper is divided into 3 parts.

Part 1 – IDS challenge in detecting Web-based attack. This part describes the characteristics of web-based attack, sample attack scenario, IDS roles and limitation, and the analysis of IDS characteristics that would be able to capture such attacks.

Part 2 – Network Detect. This part discusses three selected network detects. Two detects were based on log downloaded from www.incident.org web site and the one was the actual log, with necessary information obfuscated, from one company's network.

Part 3 – Analyse This. This part analyses log files of 5 consecutive days downloaded from www.incident.org. The log files included scan files, alert files and out-of-spec files. Event of interests were selected and the details analysis were performed. The recommendation based on found attacks were given in the executive summary.

References of each part are listed separately for the purpose of referring.

Assignment 1 - Detecting web-based attack – IDS challenge

1.1 Background

The purpose of this paper is to describe the attack at the web application level and analyse how well the intrusion detection system can detect such attacks.

E-business is another channel of offering services or products to the customers. It has become a vital piece of business or in some case, it is the whole piece of the business all by itself. To offer an e-business service, web-based infrastructure need to be developed and security measures must be in place. When you connect your infrastructure to the Internet, everyone can access the web site anytime and anywhere. One of the high risk issue facing information technology as well as management is Web application security risk, which allows attacker to bypass the firewall and evade the IDS through the legal HTTP or HTTPS requests. Results of these risks include hackers gaining unauthorised access by authentication bypass, unauthorised disclosing of customer's sensitive information, impersonating other customers or financial impact such as unauthorised money transfer. More and more attacks are carried out at the application level. The statistics (from Gartner) says 75% of today's successful attack involves Web Application. Well, this number is on the rise.

The web application risk has now become the technical challenge for the intrusion detection system, which has the main functionality to detect suspicious or attempt for all these attacks. Why didn't today IDS be able to detect all these risks and what will happen to the security in the future when there is an obvious trend that most of the client-server application will be transformed to web-based. Wouldn't this increase more risks? What would be IDS role to solve this complex and challenging problem?

1.2 Characteristics of Web-based Attack

Attack at the application level is rather different from the attack at the network level. Type of risks, vulnerabilities, and exploit techniques for these 2 levels are different. Most of the vulnerabilities at the application level can be exploited even if the infrastructure (such as host and network devices) are securely setup. These risks are unique by each application. Risk that is found with one application is likely not to appear in the other application, although common risk can be found easily and that's why it cannot be easily fixed by just installing patch or upgrade version.

Some risk can be detected with web server log, only if the attack is performed using with the manipulation of URL. Some can't be detected with web server log or web application log. As in most cases, the application is designed to log the transactions carried out in an orderly fashion but not through changing of hidden field or parameter manipulations.

1.3 Sample Attack Scenario

To give a clearer picture, the following are sample vulnerabilities that can be exploited at the application level;

Example 1 : Hidden Field Manipulation. The attacker manipulates the value in the hidden fields and sends that value back to the server. In the scenario below, an attacker is trying to buy a book online. The book will cost him $484.10. However, he noticed that the price of the book is stored as a hidden field. He tried to manipulate the value from 484.10 to 4.84 or even –4.84!

Original form

```
<form action=http://www.sample.com/book.pl method ="POST">
<input type="hidden" name="price" value="484.1">
<input type="hidden" name="product" value="Book">
<input type="hidden" name="quantity" value="1">
<input type="submit" name="submit" value="Buy Now">
</form>
```

Correct request	**Attack request**
POST /book.pl HTTP/1.0 Price=484.1	POST /book.pl HTTP/1.0 Price=**4.84**

If the application is poorly written, the edited value will be submitted to the server and get processed!

Example 2: Session Hijack. The attacker tries to impersonate the legitimate user by stealing the authenticated session. This attacker is fully aware that by brute forcing the password, the system will usually lock this user id out within 3-5 attempts. So the attacker decided to use a different way, session hijack. This could be done by stealing cookie information and inject that cookie to the server.

The Legitimate user login

```
cmd> POST /eeee/login.aspx HTTP/1.0
cmd> Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
cmd> Referer: http://test.com/eeee/login.aspx
cmd> Content-Type: application/x-www-form-urlencoded
cmd> User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; DigExt)
cmd> Host: test.com
cmd> Content-Length: 59
cmd> Cookie: SessionId=mpgwsyuadexo2c55cuhvojy5
```

The Attacker login

```
cmd> POST /eeee/login.aspx HTTP/1.0
cmd> Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
cmd> Referer: http://test.com/eeee/login.aspx
cmd> Content-Type: application/x-www-form-urlencoded
cmd> User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; DigExt)
cmd> Host: test.com
cmd> Content-Length: 21
cmd> Cookie: SessionId=qpgwsfewdewo2c00cuhvoje3
```

The Attacker impersonate the legitimate user

```
cmd> POST /eeee/login.aspx HTTP/1.0
cmd> Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
cmd> Referer: http://test.com/eeee/login.aspx
cmd> Content-Type: application/x-www-form-urlencoded
cmd> User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; DigExt)
cmd> Host: test.com
cmd> Content-Length: 21
cmd> Cookie: SessionId=mpgwsyuadexo2c55cuhvojy5
```

If there is no control at the server, the attacker could become the legitimate user and perform all kinds of activities on behalf of him.

You might wonder how cookie from one machine could be stolen by attacker from the other machine. This is easy. The attacker could use a vulnerability called 'Cross-Site Scripting' from any web site to lure the user into clicking on the link (or the button). Here're possible scenarios:

Attacker → Vulnerable Web Site	Submit a message to the discussion forum. A message will have embedded malicious script which will steal the cookie and post on the attacker's web site.
Victim → Attacker's Web Site	When the victim clicked on the link embedded in the e-mail, a malicious script will be executed and cookie will be stolen and post on the attacker's web site.

Or

Attacker → Victim	Send an e-mail to the user with embedded malicious script. An e-mail message will have interesting subject such as 'You won the lottery' or 'Special Job Offer'. This would trick the user into clicking on the e-mail.
Victim → Attacker's Web Site	When the victim clicked on the link embedded in the e-mail, a malicious script will be executed and

cookie will be stolen and post on the attacker's web site.

Example 3: Direct Access Browsing. The attacker directly access the web page without going the through the authentication. This results in the attacker being able to impersonate another user and perform illegal activities.

Following the 'segregation of duties' concept, the e-application is designed to have different levels of users performing different roles. The system is designed with 3 different roles including the Maker, the Approver and the Administrator. The Maker has the ability to create the transaction. The Approver will verify and approve the transaction created by the Maker. The Administrator perform user management role such as user ID creation, deletion and privilege management. The Administrator is not allowed to perform any financial transaction.

After authentication, both the Maker and the Approver will see different screen:

the Maker will see this URL:
http://www.company.com/create.jsp

```
With the following functions:
Create Transaction
View Transaction
```

the Approver will see this URL:
http://www.company.com/approve.jsp

```
With the following functions:
Approve Transaction
View Transaction
View Company's Balance
```

Some programmer uses file name to control level of authorization base on user group. For example, when user A login, the system knows that user A is under Maker group, the system will direct user A to http://www.company.com/create.jsp. When user B (the Approver) logs in, the system direct user B to a different path, http://www.company.com/approve.jsp.

If the control is not properly set up, the Maker could be able to impersonate the Approver by directly access http://www.company.com/approve.jsp and be able to approve the transaction created by himself/herself.

It is almost impossible for the IDS to detect the above scenarios that know that such requests are not legitimate and should be stopped.

There are more vulnerabilities at the web application waiting to be exploited. Following is a summary from OWASP:

Top vulnerabilities in Web Application [1]

A1	**Unvalidated Input**	Information from web requests is not validated before being used by a web application. Attackers can use these flaws to attack backend components through a web application.
A2	**Broken Access Control**	Restrictions on what authenticated users are

		allowed to do are not properly enforced. Attackers can exploit these flaws to access other users' accounts, view sensitive files, or use unauthorized functions.
A3	**Broken Authentication and Session Management**	Account credentials and session tokens are not properly protected. Attackers that can compromise passwords, keys, session cookies, or other tokens can defeat authentication restrictions and assume other users' identities.
A4	**Cross Site Scripting (XSS) Flaws**	The web application can be used as a mechanism to transport an attack to an end user's browser. A successful attack can disclose the end user's session token, attack the local machine, or spoof content to fool the user.
A5	**Buffer Overflows**	Web application components in some languages that do not properly validate input can be crashed and, in some cases, used to take control of a process. These components can include CGI, libraries, drivers and web application server components.
A6	**Injection Flaws**	Web applications pass parameters when they access external systems or the local operating system. If an attacker can embed malicious commands in these parameters, the external system may execute those commands on behalf of the web application.
A7	**Improper Error Handling**	Error conditions that occur during normal operation are not handled properly. If an attacker can cause errors to occur the web application does not handle, they can gain detailed system information, deny service, cause security mechanisms to fail, or crash the server.
A8	**Insecure Storage**	Web applications frequently use cryptographic functions to protect information and credentials. These functions and the code to integrate them have proven difficult to code properly, frequently resulting in weak protection.
A9	**Denial of Service**	Attackers can consume web application resources to a point where other legitimate users can no longer access or use the application. Attackers can also lock users out of their accounts or even cause the entire application to fail.
A10	**Insecure Configuration Management**	Having a strong server configuration standard is critical to a secure web application. These servers have many configuration options that affect

security and are not secure out of the box.

1.4 Causes of Web-based Attack

The major cause of the web-based attacks is from a poorly-written program. Most programmers are not aware of the security risk and are not trained to write a secure code. Even trying to keep up with the business deadline is already tough enough for the programmers, so most of them seems to ignore security. Also, security is not included in the SDLC process from the start. In many cases, security will only be considered before the system is launch, which is too late.

To assess if these risks exist in the application, penetration test or source code audit must be performed. There is no magic scanning tool that upon setting a few parameters and pressing the button, it will give you a list of the attacks possible on this application. Each scanning tool has its own limitation. Thus, manual test is still required to attack with advanced techniques.

1.5 IDS Roles

There are a number of solutions available in the market to prevent and detect some types of web application attacks such as SQL injection, URL attack, XSS or known vulnerabilities. Sample solutions are some network level firewalls, mod_security, or web application firewall.

What is the IDS role in detecting these attacks? Snort rules have a number of signatures that support the detection of web application attacks. Sample signatures are capable of detecting some web hacking attempts are web-attacks.rules, web-cgi.rules, web-client.rules, web-coldfusion.rules, web-iis.rules, web-misc.rules and web-php.rules. As of the day of writing this paper, altogether there are 1,065 rules, which account for almost 43% of total snort signatures. I strongly believe that this number is on the rise as there is a definite need for a web application IDS in the markets to specifically detect the attacks at the application level or warn the administrator such attempts. However, the network-based IDS (such as Snort) may not be as good as host-based IDS with the integration with the web server and the application framework. But how well can the host-based IDS detect such attacks considering the following factors and limitations:

- The IDS can only detect the lower layer protocols of the OSI. Although some signatures have been designed to look at the HTML tag to detect the special character or wording such as '<script>' which can be used for Cross-Site Scripting (XSS) but how could the IDS differentiate the wrong from right, like the case mentioned above how to detect that the price should be '484.1', not '4.84'.
- The IDS works very well with known vulnerabilities and patterns. The web application does not have patterns. The program is developed base on

each programmer's approach and technique. Unlike operating system or web server application, web application is unique in its fancy features, business requirements, parameters and field names.

- Field name, file name and parameters used are all different. Like the direct browsing case raised above, how could the IDS detect that the Maker accessed the Approver page.

Below are the characteristics of the IDS to addressed the above issues and be able to detect the web application attacks:

- The IDS should have knowledge of the application and understand the application's behaviour. Although this may require a lot of effort, but it's mandatory. Unlike the vulnerabilities of the operating system or the system software, the vulnerabilities at the application level are unique and are not known. The only way for the IDS to perform a good job is to understand the application's behaviour and then create appropriate rule set.
- Most IDS could read the content of the HTTP but it should have more ability to analyse HTTP method, Header Length, Header size, Header contents, URI, POST contents.
- The IDS should have the ability to analyse abnormal activities in all input fields whether they are form field, hidden field, drop-down list, radio-button or any other kinds of input. We already learned that 'all input is evil'. Invalidated input is the number one issue that causes SQL injection, cross-site scripting, and injection flaws. The IDS should always treat all input as malicious. And when I refer to input, I mean character sets, data type and input length. Users should be allowed to define user requests and input characters allowed as a rule set such as define regular expression. (For Snort, we can actually write perl script to detect the web attack by utilizing 'pcre' in payload detection rule options. PCRE allows users to write perl regular expression in snort rules [9].) Input length is as important as input type. Buffer overflow through web application is possible by changing input length and send a large number of characters to the server.
- Alert message should be clear and categorized according to the top ten risk above or follow the existing standards such as VulnXML or AVDL. Both have been developed to unify web application vulnerabilities that have been discovered. AVDL is more business-oriented while VulnXML is technical. (Please refer to web links provided in Further Reading section for more information on both standards.)
- The IDS should have the ability to minimize false positives.

We hardly see the web application IDS in the market. It could be that the vendor is still working on it. While we are waiting for the solution to become available, we could make use of the products already existing such as web application firewall. Some of the web application firewalls have some of the above features. Logs generated from the firewall represent the attack attempts at the application level and thereby could be utilized as input to the IDS.

Comment:

Comment:

NOTE- This might not be relevant to the IDS but I think it is worthwhile to note that the web application IDS or even the web application firewall should be used in addition to the secure coding practice, which need to be enforced in all development projects. I personally believe in layers of defense and also believe that the problem should be corrected at the source. If the application causes the problem, then the application needs to be fixed. And by that, I mean writing a secure code. The application IDS can be used as a mean to detect if the program is written securely.

1.6 References

[1] OWASP The Open Web Application Security Project. "The Ten Most Critical Web Application Security Vulnerabilities 2004 Update". OWASP Website. URL: http://www.owasp.org/documentation/topten (27 January 2004)

[2] Microsoft. "Improving Web Application Security, Threats and Countermeasures". Microsoft Corporation Website. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/threatcounter.asp (30 June 2003)

[3] Christopher Kruegel and Giovanni Vigna. "Anomaly Detection of Web-based Attacks". Reliable Software Group, University of California Website. URL: http://www.cs.ucsb.edu/~vigna/publications.html (October 2003)

[4] Ivan Ristic. "Introducing Mod_security". O'Reilly Media Website. URL: http://www.onlamp.com/pub/a/apache/2003/11/26/mod_security.html (26 November 2003)

[5] W3C. "Hypertext Transfer Protocol -- HTTP/1.1". World Wide Web Consortium Website. URL: ftp://ftp.isi.edu/in-notes/rfc2616.txt (June 1999)

[6] Inside the Insider Threats. Opinion by Mudge, Intrusic Inc.,

[7] Marcus Bailey, "Defeating Perimeter Security With HTTP". GSEC Website. URL: http://www.giac.org/practical/Marcus_Bailey_GSEC.doc (August 12 2002)

[8] K. K. Mookhey and Nilesh Burghate. "Detection of SQL Injection and Cross-site Scripting Attacks". SecurityFocus Website. URL: http://www.securityfocus.com/infocus/1768. (March 2004)

[9] Philip Hazel. "Perl Compatible Regular Expressions". PCRE Website. URL: www.pcre.org. (February 2004)

1.7 Further Readings

[10] OASIS. "Application Vulnerability Description Language (AVDL) Ratified as OASIS Standard". OASIS Website. URL: http://www.oasis-open.org/news/oasis_news_06_23_04.php

[11] AVDL. "Application Vulnerability Development Language FAQ" ADVL Website. URL: http://www.avdl.org/FAQ.html

[12] OWASP. "VulnXML". OWASP Website. URL: http://www.owasp.org/vulnxml

Assignment 2 – Network Detects

2.1 Detection 1

2.1.1 Source of trace

Source of trace was file 2003.12.15.6 downloaded from www.incidents.org/log.

- Statistics (from Ethereal)
  File length: 3000044
  Format: libpcap
  Start time: 2:07:16.936242
  End time: 2:08:17.518422
  Elapsed time between first and last packet: 60.582 seconds
  Packet count: 36672
  Snapshot length: 96

I used the Statistics function of ethereal to summarize a list of IP addresses and MAC addresses in the file. And I looked up a vendor Ethernet MAC address from the web site http://www.coffer.com/mac_find/. Below is an architecture base on my understanding:

```
10.10.10.165 (00:03:47:8c:89:c2 Intel machine  ) --->  3COM (00:01:02:79:91:ed) --->
Sniffer ----> 192.168.17.68 (00:50:56:40:00:6D VMWARE)
```

Observation:
File date and timestamp reported are different. The file name indicates that the data should be 2003/12/15 but the date specified in all packets actually indicated packets generated on 2003/11/19. I understand that some technique was used to obfuscate the information, such as modify ip address, as checksum of all packets are correct. Or no obfuscation has been done.

2.1.2 Detect was generated by

The file is stored in tcpdump binary format. A detect presented in this assignment was generated by Snort version 2.1.1, which I ran the analysis with my Windows 2000 Server machine. I ran snort in the NIDS mode with standard snort ruleset downloaded on 2 May 2004. All rules files were enabled. Command that was used:

```
C:\snort\bin\snort -r 2003.12.5.6 -c c:\snort\etc\snort.conf -l ex1 -X -d -A full
```

-r 2003.12.5.6	read source file 2003.12.5.6
-c c:\snort\etc\snort.conf	run against the configuration file snort.conf

-l ex1	log the output file (alert file and log file) in ex1 folder
-X	dump the raw packet data starting at the link layer (in this case, this is the Ethernet header)
-d	dump the application layer (dump the packet payloads with the packet headers)
-A full	display text alert with full packet headers

The selected alert result is as follow:

```
[**] (http_inspect) BARE BYTE UNICODE ENCODING [**]
11/19-02:08:04.823979 10.10.10.165:1085 -> 192.168.17.68:80
TCP TTL:128 TOS:0x0 ID:42592 IpLen:20 DgmLen:41 DF
***A**** Seq: 0xE4F18713  Ack: 0x16A6B6DB  Win: 0x4470  TcpLen: 20
```

The above packet looks like a response from host 10.10.10.165 to host 192.168.17.68. The snort rule that trigger the 'Bare Byte Unicode Encoding' was the http_inspect in the preprocessor configure. Preprocessors take the decoded packets from the Snort packet decoder and can examine or manipulate them before they are handed to the detection engine [1].

Note that the chosen alert has destination host running on VMware. A Vmware hold several servers. From Etherreal, the following IPs are found running:
10.10.10.1
172.20.11.52
172.20.11.80
172.20.201.198
172.20.201.2
192.168.17.129
192.168.17.135
192.168.17.68

```
preprocessor stream4_reassemble

preprocessor http_inspect: global \
   iis_unicode_map unicode.map 1252

preprocessor http_inspect_server: server default \
   profile all ports { 80 8080 8180 } \
            oversize_dir_length 500
```

Note that "profile all" includes the 'bare byte decoding' enabled. The following configuration was displayed when you run snort (without quiet option enabled). You can see that the bare byte option was set to YES.

```
HttpInspect Config:
   GLOBAL CONFIG
     Max Pipeline Requests:    0
     Inspection Type:          STATELESS
     Detect Proxy Usage:       NO
     IIS Unicode Map Filename: c:\snort\etc\unicode.map
     IIS Unicode Map Codepage: 1252
   DEFAULT SERVER CONFIG:
     Ports: 80 8080 8180
     Flow Depth: 300
     Max Chunk Length: 500000
     Inspect Pipeline Requests: YES
     URI Discovery Strict Mode: NO
     Allow Proxy Usage: NO
```

Disable Alerting: YES
Oversize Dir Length: 500
Only inspect URI: NO
Ascii: YES alert: NO
Double Decoding: YES alert: YES
%U Encoding: YES alert: YES
Bare Byte: YES alert: YES
Base36: OFF
UTF 8: OFF
IIS Unicode: YES alert: YES
Multiple Slash: YES alert: NO
IIS Backslash: YES alert: NO
Directory: YES alert: NO
Apache WhiteSpace: YES alert: YES
IIS Delimiter: YES alert: YES
IIS Unicode Map: GLOBAL IIS UNICODE MAP CONFIG
Non-RFC Compliant Characters: NONE

To look into the packet in details, I used windump to generate packet in hex format:

```
C:\windump>windump -r 2003.12.15.6 -x -vv -n "dst host 192.168.17.68 and port 80"

02:08:04.823979 IP (tos 0x0, ttl 128, id 42592, len 41) 10.10.10.165.1085 > 192.
168.17.68.80: . [tcp sum ok] 3841034003:3841034004(1) ack 380024539 win 17520 (DF)
                         4500 0029 a660 4000 8006 6dd3 0a0a 0aa5
                         c0a8 1144 043d 0050 e4f1 8713 16a6 b6db
                         5010 4470 b6b3 0000 9000 0000 0000
```

There's 1-byte data sent over. From the data field displayed in hex above, it's '90' (NOP byte). NOP is usually used to pad the TCP options as TCP options must fall on a 4-byte boundaries. If they are less than 4-bytpes, NOP will be used to pad. NOP could also be used to DoS the target host. From the alert, it looks like host 10.10.10.165 is trying buffer overflow on host 192.168.17.68. But let analyse further if this is a false positive.

Now let's also look at other packets associated with host 192.168.17.68 for a better analysis and to determine possible role of this host.

```
C:\Snort\log\old>snort -r 2003.12.15.6 -v -q "host 192.168.17.68"
11/19-02:07:48.841453 10.10.10.165:1691 -> 192.168.17.68:1080
TCP TTL:128 TOS:0x0 ID:41554 IpLen:20 DgmLen:48 DF
******S* Seq: 0x293AB444  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/19-02:07:51.828714 10.10.10.165:1691 -> 192.168.17.68:1080
TCP TTL:128 TOS:0x0 ID:41720 IpLen:20 DgmLen:48 DF
******S* Seq: 0x293AB444  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/19-02:07:57.968183 10.10.10.165:1691 -> 192.168.17.68:1080
TCP TTL:128 TOS:0x0 ID:42089 IpLen:20 DgmLen:48 DF
******S* Seq: 0x293AB444  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/19-02:07:59.094302 10.10.10.165:1703 -> 192.168.17.68:1080
TCP TTL:128 TOS:0x0 ID:42248 IpLen:20 DgmLen:48 DF
******S* Seq: 0x296B9F43  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
```

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

11/19-02:08:02.163499 10.10.10.165:1703 -> 192.168.17.68:1080
TCP TTL:128 TOS:0x0 ID:42416 IpLen:20 DgmLen:48 DF
******S* Seq: 0x296B9F43  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/19-02:08:04.823979 10.10.10.165:1085 -> 192.168.17.68:80
TCP TTL:128 TOS:0x0 ID:42592 IpLen:20 DgmLen:41 DF
***A**** Seq: 0xE4F18713  Ack: 0x16A6B6DB  Win: 0x4470  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/19-02:08:08.302980 10.10.10.165:1703 -> 192.168.17.68:1080
TCP TTL:128 TOS:0x0 ID:42605 IpLen:20 DgmLen:48 DF
******S* Seq: 0x296B9F43  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/19-02:08:09.095015 10.10.10.165:1711 -> 192.168.17.68:1080
TCP TTL:128 TOS:0x0 ID:42615 IpLen:20 DgmLen:48 DF
******S* Seq: 0x2999ACB6  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/19-02:08:12.089067 10.10.10.165:1711 -> 192.168.17.68:1080
TCP TTL:128 TOS:0x0 ID:42647 IpLen:20 DgmLen:48 DF
******S* Seq: 0x2999ACB6  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

Run time for packet processing was 0.750000 seconds
```

The above information is still not enough to analyse the intruder's attempt so I have merged all log files into one with the following command.

C:> mergecap -w merge 2003.12.15.1 2003.12.15.2 2003.12.15.3 2003.12.15.4 2003.12.15.5 2003.12.15.6 2003.12.15.7 2003.12.15.8 2003.12.15.9 2003.12.15.10 2003.12.15.11 2003.12.15.12 2003.12.15.13 2003.12.15.14

Then I ran snort again with the same set of rules. Some of the results that I got are shown below:

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

11/19-02:05:05.114599 10.10.10.165:2695 -> 192.168.17.68:1
TCP TTL:128 TOS:0x0 ID:21572 IpLen:20 DgmLen:48 DF
******S* Seq: 0x8158842  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/19-02:05:05.115021 10.10.10.165:2696 -> 192.168.17.68:2
TCP TTL:128 TOS:0x0 ID:21573 IpLen:20 DgmLen:48 DF
******S* Seq: 0x816111F  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/19-02:05:05.116627 10.10.10.165:2697 -> 192.168.17.68:3
TCP TTL:128 TOS:0x0 ID:21574 IpLen:20 DgmLen:48 DF
******S* Seq: 0x816B61C  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/19-02:05:05.117195 10.10.10.165:2698 -> 192.168.17.68:4
TCP TTL:128 TOS:0x0 ID:21575 IpLen:20 DgmLen:48 DF
******S* Seq: 0x817B1FC  Ack: 0x0  Win: 0x4000  TcpLen: 28
```

```
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/19-02:05:05.117646 10.10.10.165:2699 -> 192.168.17.68:5
TCP TTL:128 TOS:0x0 ID:21576 IpLen:20 DgmLen:48 DF
******S* Seq: 0x81865D4  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Above are just sample packets. There are long list of scan packets originating from 10.10.10.165. We can tell from the generated packets that the attacker did TCP port scan and UDP port scan to identify running ports of host 192.168.17.68. Host 192.168.17.68 responded to the following ports ftp, telnet, ftp-data and ssh with RST/ACK which means port are not listening or blocked by the firewall/router. Host 192.168.17.68 responded to http and https with SYN/ACK which means 80 and 443 are running. At this point, the attacker knew that this is a web server. So the attacker was trying to scan the server with, probably, web server scanning tools.

2.1.3 Probability the source address was spoofed

HTTP session requires a complete 3-way handshake. The data payload, as displayed in hex above (section 2.1.2), indicated that this packet has most likely completed a handshake. In addition, the source host performed the fingerprint of the destination host through various scans, the source host need the result to determine which ports are running. Therefore, it is unlikely that the source ip would be spoofed.

2.1.4 Description of attack

Bare byte encoding is an IIS trick that uses non-ASCII chars as valid values in decoding UTF-8 values. This is NOT in the HTTP standard, as all non-ASCII values have to be encoded with a %. Bare byte encoding allows the user to emulate an IIS server and interpret non-standard encodings correctly. There are no legitimate clients that encoded UTF-8 this way, since it is non-standard [1].

For more descriptions on the terms being used,

Unicode is a single unified character set. Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language [5].
UTF-8 is a method to encode character to Unicode. (one of the three common encoding method) UTF-8 encodes each Unicode character as a variable number of 1 to 4 octets. Using Unicode/UTF-8, you can write in emails and source code things such as Mathematics and Sciences or different languages [6].
ASCII - ASCII code is the numerical representation of a character. Because computer can only understand numbers. All non-ASCII characters usually screwed up when output it to the browser, such as some special character or language. UTF-8 supports ASCII characters but not very good in non-ASCII character [8].

A NOP shown in the alert is an instruction that will tell the CPU to do nothing and wait for the next command. If there are large chunks of NOP data, it will be that someone is trying a buffer overflow on the web server with the No Operation instructions. But for this case, a single NOP is not enough to overflow the server. So I think this could be some old encoding that trigger snort rule. The source host sent the packet contains encoding character which trigger snort rule to alert.

2.1.5 Attack mechanism

As I analyzed the combined log, I found that host 10.10.10.165 tried various reconnaissance against host 192.168.17.68 including TCP scan, UDP scan, socks scan and probably Unicode attack as well. Usually I would try to obtain the correlation evidence from the secondary resource such as web server to confirm the attack. However, the secondary resource (such as web server log, firewall log) is not available in this case and we only have the snort log less than 2 hours. I have tried drawing the attack scenario base on the limited information.

Host 10.10.10.165 perform port scans against 192.168.17.68. When port 80 is found running, host 10.10.10.165 used web server scanner to specifically scan vulnerabilities of the web server. As part of the scanning process, one of the scanning policies happened to use the non-ASCII character that trigger the snort rule to alert.

To confirm my belief, I set up a Web server at home and wrote a few e-commerce pages. Then I ran snort with the same rule set and used N-Stealth, a HTTP Security Scanner, to scan the e-commerce application that I created. N-Stealth used various combinations of possible web application attack and generated a number of alerts including bare byte Unicode encoding alerts. When looking at packets in general, I noticed the similarity. A number of alerts showing port scanning were generated and among these alerts is the bare byte Unicode encoding alert, which had similar characteristics, such as a response packet (TCP Flag A) and 1-byte data (NOP). So this is not the DoS.

2.1.6 Correlations

I could not find the analysis on bare byte Unicode encoding from the previous assignments so I did some research on the Internet. Bare byte Unicode encoding is considered under the Unicode attack category. The Unicode attack was previously raised by Bruce Schneier at http://www.schneier.com/crypto-gram-0007.html [4]

The only paper that chose to analyse this alert is Blaine Hein's GCIA paper [9]. Blaine explained in his paper that the omission of the HTTP method triggers the rule "Bare Byte Unicode Encoding". This is just part of the stream of data being sent to the web server. As Snort is currently stateless, the HTTP analysis is currently only performed on a per packet basis. On its own this is an HTTP packet that does not conform to the http standard in that the data field does not begin with a HTTP method, and instead begins with Unicode bytes.

2.1.7 Evidence of active targeting

The attacker was trying to scan for vulnerable host. Host 192.168.17.68 is not the key target at first. But it is actively targeted later.

2.1.8 Severity

Severity = (criticality + lethality) (system countermeasures + network countermeasures)

Criticality 5: This target is a web server. Although I do not have enough information on what kind of information presented on this web server but given the fact that https is used so the information must be important. I would rate this 5.

Lethality 0: Although the alert is generated among the reconnaissance, but this alert itself is not an attack.

System countermeasures 4: The server did not seem to response to port scans. Unnecessary ports may have been removed. Although port 80 and 443 are wide opened, this is normal for web server.

Network countermeasures 2: No evidence if firewall is running so I give 2 for this.

Severity = (5 + 0) - (4 + 2) = -1

2.1.9 Defensive recommendation

- Apply necessary patches to prevent known vulnerabilities at the web server or operating system.
- Apply a secure programming concept when developing the web application. Necessary input validation must be in place to filter out characters that will not be needed. For most of the case, non-ASCII character won't be needed for any field.
- Have network measures in place such as properly configured router, firewalls and IDS.

2.1.10 Multiple choice questions

Which of the following statements about NOP is true?

A. NOP is a mandatory field in IP datagram.
B. NOP is used to pad TCP options.
C. If NOP is found in any IP datagram, it shows that the system has been compromised.
D. NOP is the method used to DoS the target host.

Answer: B

2.1.11 References

[1] The Snort Project. Snort Users Manual 2.1.2. (March 2004)
[2] Benjamin D. Thomas. "IDS Evasion with Unicode". Linux Security Website. URL: http://www.linuxsecurity.com/articles/intrusion_detection_article-2231.html (Jan 2001)
[3] Daniel J. Roelker, "HTTP IDS Evasions Revisited". IDS Research Website. URL: http://docs.idsresearch.org/http_ids_evasions.pdf.
[4] Bruce Schneier. "Security Risks of Unicode". Crypto-Gram Newsletter, Bruce Schneier Website. URL: http://www.schneier.com/crypto-gram-0007.html. (July 2000)
[5] The Unicode Consortium. "What is Unicode?" Unicode Website. URL: http://www.unicode.org/standard/WhatIsUnicode.html
[6] Markus Kuhn. "UTF-8 and Unicode FAQ for Unix/Linux". The Computer Laboratory, Universitoy of Cambridge Website. URL: http://www.cl.cam.ac.uk/~mgk25/unicode.html (June 2004)
[7] W3C. "Hypertext Transfer Protocol -- HTTP/1.1". World Wide Web Consortium Website. URL: ftp://ftp.isi.edu/in-notes/rfc2616.txt (June 1999)
[8] W3C. "HTML Document Representation". World Wide Web Consortium Website. URL: http://www.w3.org/TR/REC-html40/charset.html
[9] Blaine Hein. "GCIA Practical Assignment". GIAC Website. URL: http://www.giac.org/practical/GCIA/Blaine_Hein_GCIA.pdf (May 2004)

2.2 Detection 2

2.2.1 Source of trace

A large company with medium-sized network has never implemented the intrusion detection in their environment before. A company has a corporate security policy but has never verified if the policy has been followed by the staff. I installed snort with proper permission from the IT manager and top management of the company. Snort was set up in the test machine placed in one zoning in the production environment. Snort was run for about half an hour to detect the intrusion attempts. The purpose of this test is to demonstrate to the management of possible attacks or violation of security policy that took place in the network. It is note that permission to present the result presented in this paper has been granted with the condition that ip addresses obfuscated and identity not disclosed.

- Statistics (from Ethereal)

  File length: 164837
  Format: libpcap
  Start time: 2004-5-19 15:36:28.986858
  End time: 2004-5-19 16:21:19.997928
  Elapsed time between first and last packet: 2691.011 seconds
  Packet count: 785
  Snapshot length: 1514

```
External Network → Internet -> Router -> Firewall -> Web Server
                                                → Switch → Snort → 192.168.x.x (Hosts)
```

To preserve the confidentiality, MAC addresses, vendor products and details zoning information will not be presented here.

2.2.2 Detect was generated by

I installed snort on the Windows 2003 machine. This machine is located on the internal network as shown in the picture above. Snort was configured with rule downloaded on 2 May 2004. All rules files were enabled. The following command was used to capture the traffic and to generate alerts:

```
C:\snort\bin\snort -c c:\snort\etc\snort.conf -l log -X -d -A full
```

-c c:\snort\etc\snort.conf	run against the configuration file snort.conf
-l log	log the output file (alert file and log file) in log folder
-X	dump the raw packet data starting at the link layer (in this case, this is the Ethernet header)
-d	dump the application layer (dump the packet payloads with the packet headers)
-A full	display text alert with full packet headers

There were a number of alerts generated but the selected ones for this analysis is NETBIOS SMB-DS IPC$ share unicode access.

```
[**] [1:2466:1] NETBIOS SMB-DS IPC$ share unicode access [**]
```

```
[Classification: Generic Protocol Command Decode] [Priority: 3]
05/19-15:36:28.986858 192.168.0.78:4607 -> 192.168.0.151:445
TCP TTL:128 TOS:0x0 ID:13830 IpLen:20 DgmLen:120 DF
***AP*** Seq: 0x3B96572A  Ack: 0x529281A8  Win: 0xF922  TcpLen: 20
```

which was triggered by the following signature:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"NETBIOS SMB-DS IPC$ share unicode
access"; flow:to_server,established; content:"|00|"; depth:1; content:"|FF|SMBu"; depth:5;
offset:4; byte_test:1,>,127,6,relative; content:"I|00|P|00|C|00 24 00 00|"; distance:32;
nocase; classtype:protocol-command-decode; sid:2466; rev:3;)
```

This rule will be triggered for any TCP packets to external destination host via port 445 and the packet content the SMB share name.

2.2.3 Probability the source address was spoofed

Source address is likely not to be spoofed. These addresses are internal address being used in the actual corporate environment and the attacker need response to the request to access shared information in the other host. Note that to support this conclusion, I have requested for more information from the IT team and have already mapped the MAC to actual hosts to check if this is indeed the source of this traffic.

2.2.4 Description of attack

NETBIOS SMB-DS IPC$ share unicode access: This event is generated when an attempt is made to gain access to private resources in Windows machine. Some internal staff shared the folder in his machine to other users in the network. It doesn't seem to harm the network in terms of bandwidth but it is very dangerous when considering that this could be a great channel to spread virus, Trojans and worm. Also, this put the confidentiality of corporate information at risk. The user could transfer company top secret information to external unauthorised host. This could be considered as attack against information asset. It is easy and can be done without requiring any tools [4].

2.2.5 Attack mechanism

The alert came in pattern. Following is a pattern before share is detected.

```
[**] [1:466:1] ICMP L3retriever Ping [**]
[Classification: Attempted Information Leak] [Priority: 2]
05/19-15:38:30.582676 192.168.0.78 -> 192.168.0.156
ICMP TTL:32 TOS:0x0 ID:15402 IpLen:20 DgmLen:60
Type:8  Code:0  ID:512   Seq:28161  ECHO
[Xref => http://www.whitehats.com/info/IDS311]

[**] [1:408:4] ICMP Echo Reply [**]
```

```
[Classification: Misc activity] [Priority: 3]
05/19-15:38:30.582917 192.168.0.156 -> 192.168.0.78
ICMP TTL:128 TOS:0x0 ID:21037 IpLen:20 DgmLen:60
Type:0  Code:0  ID:512  Seq:28161  ECHO REPLY

[**] [1:2466:1] NETBIOS SMB-DS IPC$ share unicode access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
05/19-15:38:59.735989 192.168.0.78:4637 -> 192.168.0.156:445
TCP TTL:128 TOS:0x0 ID:15447 IpLen:20 DgmLen:130 DF
***AP*** Seq: 0x3DBAA6F7  Ack: 0x937E9B92  Win: 0xF91D  TcpLen: 20
```

Host 192.168.0.78 send ICMP Echo request to the destination machine, host 192.168.0.156, to determine if it's alive. The first alert is usually generated from a host running the L3 "Retriever 1.5" security scanner. From the architecture of the network, ping is allowed as there is no internal firewall to prevent this. Once ICMP packet is sent, the destination host which is alive will response with ICMP ECHO REPLY (Ping triggered snort to alert as well) within a few seconds. This is a normal ICMP stimulus and response which sometimes also used to legitimately troubleshoot networking problems. Host 192.168.0.78 learned that the target host is alive. He then tried to see if there are any shared folders and tried access them. The last alert has TCP flag set to AP, which means some data

As I informed the IT manager of the possibility that some of the staff could have violated the company's policy. The IT Manager ran a check and discovered that that some staff did really share a number of folders in his machine and some were not aware that the machine was shared. That folder contains music and various executable files. Before this could lead to the worse issue and be the source to distribute virus and Trojans, that machine was ordered to unshared the folders immediately.

To look into other packets and detect other activities performed by this staff, I used windump to filter alert from this source ip address and discovered that there were a number of alerts generated.

As I read a full packets in details, I could see the physical path name used to access the shared folders. I wrote down the information and compiled a list of share names available in any machines detected by Snort. My good guess from the share name was some folders were created to share the files for entertainment purpose. However some folders were created to share corporate information, which I had no idea what are those information but it looked interesting. I gave a full list of the folders to the IT Manager and the team and let them try. (The reason being this is considered as penetration test and according to the agreed scope, I'm not authorized to perform such test.) Some folders were not password protected and could be accessed with full privileges. The team tried to confirm this by creating a blank folder and a little notepad file to ensure that they really had unlimited access to those folders and they succeeded.

And for the corporate information that were stored in the folder, we found that those are the files generated from the accounting system and could be used for management report preparation. However, this is beyond my interest to perform further investigation if the generation of the file was beyond that person's job responsibilities.

I was requested to summarize a full list of source IP that access those folders. I ran Windump to retrieve the packets with IP 192.168.0.78 who seems to try access various shared folders available and any destination IP involved. I then gave a complete list to the IT manager for further analysis and investigation.

2.2.6 Correlations

This is the traffic detected from one company's network and it was the first time intrusion detection system was installed. Although there is no previous analysis for this alert, I did some research on the Internet and discovered that there are worms that seek file sharing and attempts to make connection to the ADMIN$ and IPC$ shares. They will then spread themselves to the remote machine via share. Sample of these worms are W32/Randin.worm.gen, W32.Netspree.Worm, W32/MoFei.worm.

2.2.7 Evidence of active targeting

Host 192.168.0.78 is the source of most alerts. But there is no clear target. I think that the target is any hosts that 192.168.0.78 could compromise. These hosts could be used as agent to perform DDoS attack to external server or even internal server.

2.2.8 Severity

Severity = (criticality + lethality) (system countermeasures + network countermeasures)

Criticality 4: The attack target the desktop. The information confidentiality of this organization is at risk. The internal host could be used as a channel to distribute virus, worm and Trojan.

Lethality 4: Although this is the confidentiality attack, the attacker could try something more severe such as Trojan plan, DoS or work attack. I would have given the rating 5 but I do not know how well the server is protected so I will give this 4.

System countermeasures 3: Apply patch against the operating system, update virus signature and turn off file sharing.

Network countermeasures 3: The network has firewall running but the firewall is placed between the external network and internal network so it is not able to prevent this. However, IDS could be used to detect the sharing.

Severity = (4 + 4) - (3 + 3) = 2

2.2.9 Defensive recommendation

- Although we do not have enough information on the attack against the company's server (as Snort was run for about half an hour only), it is recommended that the investigation be performed against company's servers to look for sign of compromises.
- Perform vulnerability scanning and system hardening of all hosts.
- Uninstall p2p clients/servers in all client machines.
- Perform inventory check of what are other unauthorized software/tools installed in the corporate machines.
- Consider using desktop management to better manage the clients and prevent them from installing unauthorized software,
- Install internal firewall and place critical servers in a separate zone. Implement proper security measures to protect the critical zone.
- Install Intrusion Detection System to detect dangerous traffic.
- Enforce corporate security policy and penalize staff who violates the security policy.

2.2.10 Multiple choice questions

What could be the result of using Gnutella p2p?
A. Violation of copyright.
B. Source to distribute Trojan, worm and virus.
C. Backdoor to gain access to internal LAN
D. All of the above

Answer: D

2.2.11 References

1. Declan Murphy, Jarlath Kelly, Keith Curley, John Vickery, and Dan O'Keeffe. "P2P Network Security". Networks and Telecommunications

Research Group Website. URL:http://ntrg.cs.tcd.ie/undergrad/4ba2.02-03/p10.html (March 2003)
2. Sandra Underhill. "Is Gnutella a Security Risk to Your Files?". Ifinisource Website. URL: http://www.infinisource.com/features/gnutella.html. (February 2001)
3. Gene R Gomez. "P2P GNUTella client request". Snort Website. URL:http://www.snort.org/snort-db/sid.html?sid=557
4. Brian Caswell, Nigel Houghton. "NETBIOS SMB-DS IPC$ share unicode access". Snort Website. URL:http://www.snort.org/snort-db/sid.html?sid=2466
5. McAfee. "W32/Deloder.worm" http://vil.nai.com/vil/content/v_100127.htm
6. Symantec. "W32.Netspree.Worm" http://securityresponse.symantec.com/avcenter/venc/data/w32.netspree.worm.html (January 2003)
7. McAfee. "W32/MoFei.worm" http://vil.mcafeesecurity.com/vil/content/v_100357.htm (June 2003)

2.3 Detection 3

2.3.1 Source of trace

Source of trace was file 2003.12.15.12 downloaded from www.incidents.org/log. The following is a statistics of the file summarized by Ethereal.

File length: 3000051
Format: libpcap
Start time: 2:17:47.123087
End time: 2:22:21.448353
Elapsed time between first and last packet: 274.325 seconds
Packet count: 34011
Snapshot length: 96

MAC addresses and IP addresses involved were summarized with the conversation function in Ethereal. I did a search of the involved product from http://www.coffer.com/mac_find/. Below is an architecture base on my understanding:

```
172.20.201.1 (00:50:56:40:00:6d – Vmware on 10.10.10.1) → Snort → 10.10.10.165
(00:03:47:8c:89:c2)
```

Host 172.20.201.1 is running on VMware which is installed on host 10.10.10.1. From the conversation list (below), 10.10.10.1 is talking to 10.10.10.165, so I think that the whole architecture is probably running on the same network.

```
Address A,Address B,Packets,Bytes,Packets A->B,Bytes A->B,Packets A<-B,Bytes A<-B,
10.10.10.165,10.10.10.1,13140,1033239,12073,954483,1067,78756,
10.10.10.228,10.10.10.1,6518,489481,3303,260157,3215,229324,
10.10.10.195,10.10.10.1,3981,242870,1976,122510,2005,120360,
10.10.10.224,10.10.10.1,2110,129328,1976,118560,134,10768,
10.10.10.1,10.10.10.141,1664,113280,7,420,1657,112860,
10.10.10.147,10.10.10.1,1508,144276,837,71402,671,72874,
10.10.10.234,10.10.10.1,1291,125860,653,54676,638,71184,
10.10.10.142,10.10.10.1,799,436152,370,27322,429,408830,
10.10.10.160,10.10.10.1,737,68454,410,34550,327,33904,
10.10.10.186,10.10.10.1,186,22804,97,8889,89,13915,
10.10.10.212,10.10.10.1,185,15336,95,6225,90,9111,
10.10.10.232,10.10.10.1,147,18780,89,7630,58,11150,
```

The selected alert is the initial communication from Master (10.10.10.165) to the Daemon (172.20.201.1).

2.3.2 Detect was generated by

The file is stored in tcpdump binary format. The detect presented in this assignment was generated by Snort version 2.1.1, which I ran the analysis with my Windows 2000 Server machine. I ran snort in the NIDS mode with standard snort ruleset downloaded on 2 May 2004. All rules files were enabled. Command that was used:

```
C:\snort\bin\snort -r 2003.12.5.12 -c c:\snort\etc\snort.conf -l ex3 -X -d -A full
```

-r 2003.12.5.12	read source file 2003.12.5.12
-c c:\snort\etc\snort.conf	run against the configuration file snort.conf
-l ex3	log the output file (alert file and log file) in ex3 folder
-X	dump the raw packet data starting at the link layer (in this case, this is the Ethernet header)
-d	dump the application layer (dump the packet payloads with the packet headers)
-A full	display text alert with full packet headers

The selected alert result is as follow:

```
[**] [1:237:2] DDOS Trin00 Master to Daemon default password attempt [**]
[Classification: Attempted Denial of Service] [Priority: 2]
11/19-02:17:52.078334 10.10.10.165:31335 -> 172.20.201.1:27444
UDP TTL:128 TOS:0x0 ID:23805 IpLen:20 DgmLen:39
Len: 11 [Xref => http://www.whitehats.com/info/IDS197]
```

The snort rule that trigger this alert is:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 27444 (msg:"DDOS Trin00 Master to Daemon default
password attempt"; content:"l44adsl"; reference:arachnids,197; classtype:attempted-dos;
sid:237; rev:2;)
```

The signature will be triggered if there's any udp traffic originate from external to internal host via port 27444 and the content contains '144adsl' [1]. This is considered as one part of the distributed denial of service category. A complete attack includes communication from Attacker → Master → Daemon → Victim. More details will be explained in the following sections. Actually the traffic can be originated externally or internally. If the traffic is generated externally, then one of the internal hosts has been compromised and used as a Daemon. If the traffic is generated internally, then one of the internal hosts could be used as a Master.

Following is a complete packet with data information in hex. Host 10.10.10.165 could be the Trin00 master and host 172.20.201.1 is the daemon. The Trin00 master communicates with the daemon via port 27444 with a string of "l44adsl" in the payload. This string is the default password for the daemon. 'Png 144adsl' is a ping command Trin00 master send to the check active daemon. This is not an attack yet, just the communication to check the daemon.

```
[**] DDOS Trin00 Master to Daemon default password attempt [**]
11/19-02:17:52.078334 10.10.10.165:31335 -> 172.20.201.1:27444
UDP TTL:128 TOS:0x0 ID:23805 IpLen:20 DgmLen:39
Len: 11
0x0000: 00 50 56 40 00 6D 00 03 47 8C 89 C2 08 00 45 00  .PV@.m..G.....E.
0x0010: 00 27 5C FD 00 00 80 11 54 04 0A 0A 0A A5 AC 14  .'\.....T.......
0x0020: C9 01 7A 67 6B 34 00 13 47 CF 70 6E 67 20 6C 34  ..zgk4..G.png l4
0x0030: 34 61 64 73 6C 00 00 00 00 00 00 00              4adsl.......

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

I looked for associating packets with these 2 IP addresses. The following packets are printed from Ethereal:

```
No.     Time                        Source                Destination           Protocol
Info
   1343 2003-11-18 14:17:52.078334 10.10.10.165          172.20.201.1          UDP
Source port: 31335  Destination port: 27444
   1344 2003-11-18 14:17:52.079597 172.20.201.1          10.10.10.165          ICMP
Destination unreachable
   1345 2003-11-18 14:17:52.091381 10.10.10.165          172.20.201.1          ICMP
Echo (ping) request
   1346 2003-11-18 14:17:52.092282 10.10.10.1            10.10.10.165          ICMP
Time-to-live exceeded
   1747 2003-11-18 14:17:53.592168 10.10.10.165          172.20.201.1          UDP
Source port: 8048  Destination port: 33222
   1749 2003-11-18 14:17:53.616513 10.10.10.1            10.10.10.165          ICMP
Time-to-live exceeded
   2066 2003-11-18 14:17:55.093657 10.10.10.165          172.20.201.1          ICMP
Echo (ping) request
   2067 2003-11-18 14:17:55.096399 172.20.201.1          10.10.10.165          ICMP
Echo (ping) reply
```

#1343 shows the chosen alert packet. #1344 shows that the daemon machine returns UDP request with host unreachable message. #1345, Master sent a ping echo request to Daemon with TTL=1, Daemon returns with message saying TTL exceeded. This could be the traceroute command that Master use to track network path to the Daemon machine. Traceroute sets TTL=1 and waits for TTL exceeded response, which will return with source IP. #1346, 10.10.10.1 (VMware that host

172.20.201.1) returned with exceeded response message, Type 11 Code 0. #2066 Master requested with TTL=2 and #2067 Daemon replied with Echo Reply, Type 0 Code 0, meaning the traceroute made its way to the Daemon machine. The Daemon machine is alive.

I reviewed other packets and alerts to look for sign of success communication between the different pairs below:

Attacker to Master (via tcp port 27665)
Master to Daemon (via udp port 27444)
Daemon to Master (via udp port 31335)

I searched for packets with port 27665 and found the following:

```
No.     Time                        Source                Destination
Protocol Info
▪ 30134 2003-11-18 14:17:28.244553 10.10.10.165          172.20.201.1          TCP
  3712 > 27665 [SYN] Seq=0 Ack=0 Win=16384 Len=0 MSS=1460
▪ 30135 2003-11-18 14:17:28.246750 172.20.201.1          10.10.10.165          TCP
  27665 > 3712 [RST, ACK] Seq=0 Ack=0 Win=0 Len=0
▪ 30290 2003-11-18 14:17:28.735780 10.10.10.165          172.20.201.1          TCP
  3712 > 27665 [SYN] Seq=0 Ack=0 Win=16384 Len=0 MSS=1460
▪ 30291 2003-11-18 14:17:28.737459 172.20.201.1          10.10.10.165          TCP
  27665 > 3712 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
▪ 30521 2003-11-18 14:17:29.247422 10.10.10.165          172.20.201.1          TCP
  3712 > 27665 [SYN] Seq=0 Ack=0 Win=16384 Len=0 MSS=1460
▪ 30522 2003-11-18 14:17:29.249234 172.20.201.1          10.10.10.165          TCP
  27665 > 3712 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
```

It looks like host 10.10.10.165 was trying to initiate connection with 172.20.201.1 via port 27665 but Host 172.20.201.1 did not response. The content data also did not contain any password that indicate that this is part of the Trin00 Attacker connect to Master (default passwordl "gOrave", default startup password "betaalmostdone", or default mdie password "killme").

I also looked for response from Daemon to Master but couldn't find any packets so I did more search, this could be the possible installation of rootkit. Rootkit could be used to conceal malicious programs and communications. I did a search for rootkit on port 1524 (also search for other rootkits on 511, 2555, 33567, 33568, 47017, 60008) but host 172.20.201.1 denied all SYN request packets to those ports.

```
11/19-02:19:01.962752 10.10.10.228:33513 -> 172.20.201.1:1524
TCP TTL:64 TOS:0x0 ID:21348 IpLen:20 DgmLen:60 DF
******S* Seq: 0x937F6B1E  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 342138 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/19-02:19:02.027226 172.20.201.1:1524 -> 10.10.10.228:33513
TCP TTL:63 TOS:0x0 ID:0 IpLen:20 DgmLen:40 DF
***A*R** Seq: 0x0  Ack: 0x937F6B1F  Win: 0x0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

It is noted that Host 10.10.10.165 also tried to contact other possible Trin00 Daemon but not successful. See packets below. (from Ethereal)

```
No.     Time        Source                Destination         Protocol Info
10091 68.610258   10.10.10.165          172.20.201.135        UDP      Source port: 31335
Destination port: 27444

Frame 10091 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:03:47:8c:89:c2, Dst: 00:50:56:40:00:6d
Internet Protocol, Src Addr: 10.10.10.165 (10.10.10.165), Dst Addr: 172.20.201.135
(172.20.201.135)
User Datagram Protocol, Src Port: 31335 (31335), Dst Port: 27444 (27444)
Data (11 bytes)

0000  70 6e 67 20 6c 34 34 61 64 73 6c                  png l44adsl

No.     Time        Source                Destination         Protocol Info
10092 68.612655   172.20.201.135        10.10.10.165          ICMP     Destination
unreachable

Frame 10092 (81 bytes on wire, 81 bytes captured)
Ethernet II, Src: 00:50:56:40:00:6d, Dst: 00:03:47:8c:89:c2
Internet Protocol, Src Addr: 172.20.201.135 (172.20.201.135), Dst Addr: 10.10.10.165
(10.10.10.165)
Internet Control Message Protocol

No.     Time        Source                Destination         Protocol Info
11820 96.037656   10.10.10.165          172.20.201.198        UDP      Source port: 31335
Destination port: 27444

Frame 11820 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:03:47:8c:89:c2, Dst: 00:50:56:40:00:6d
Internet Protocol, Src Addr: 10.10.10.165 (10.10.10.165), Dst Addr: 172.20.201.198
(172.20.201.198)
User Datagram Protocol, Src Port: 31335 (31335), Dst Port: 27444 (27444)
Data (11 bytes)

0000  70 6e 67 20 6c 34 34 61 64 73 6c                  png l44adsl

No.     Time        Source                Destination            Protocol Info
11821 96.040836   172.20.201.198        10.10.10.165          ICMP     Destination
unreachable

Frame 11821 (81 bytes on wire, 81 bytes captured)
Ethernet II, Src: 00:50:56:40:00:6d, Dst: 00:03:47:8c:89:c2
Internet Protocol, Src Addr: 172.20.201.198 (172.20.201.198), Dst Addr: 10.10.10.165
(10.10.10.165)
Internet Control Message Protocol
```

From the analysis above, host 10.10.10.165 is the Trin00 Master but host 172.20.201.1 is not the Trin00 Daemon.

2.3.3 Probability the source address was spoofed

Usually the source ip of UDP packet is easily spoofed. But in this case, the source ip is the internal host and it needs response from the destination. The Daemon need to send udp packet back to the master so that the master can collect a list of listening Daemon. So it is unlikely that the source address will be spoofed.

No packets from 172.20.201.1 to 10.10.10.165 were found. (At least not from the downloaded files). Master doesn't receive any udp packet back.

2.3.4 Description of attack

Trin00 is a tool to launch DDoS. [3] A Trin00 network looks like this: Attacker → Master → Daemon → Victim. Communication from Attacker to Master, Master to Daemon and Daemon to Master are via port 27665, 27444 and 31335 respectively.

The DDoS attack usually starts with Attacker communicate with Master via tcp port 27665, with default Trin00 password 'betaalmostdone'. Then the Master send the command to the Daemon via udp port 27444 with the default daemon password '144adsl'. The Daemon will respond to masters on udp port 31335. Master will then compile a list of listening daemons by looking for '*HELLO' in the udp response packets.

Attackers can send a number of commands to masters [3]. Examples are:

- quit - to logoff from the master
- dos IP - to launch a DDos attack against the address IP
- mdos - to launch a multiple DDos attack
- bcast - to form a list of started daemons

Masters can send commands to daemons according to what the attacker has ordered [3]. For example:

- aaa password IP - Dos attack address IP by sending UDP packets to random (0-65534) UDP ports.
- bbb password N - Period of time in seconds to run Dos attack.
- rsz N - Set size of UDP packets to N bytes.
- d1e - Shutdown the daemon

2.3.5 Attack mechanism

Host 10.10.10.165, a Trin00 Master initiated connection against 172.20.201.1 via port 27444 with default Trin00 password '144adsl' in the payload. The purpose is to check if host 172.20.201.1 is an active Daemon. This connection caused snort to alert. Host 172.20.201.1 did not response to the request. From the analysis above (refer 2.3.2) host 10.10.10.165 is the Trin00 Master but host 172.20.201.1 is not the Trin00 Daemon.

2.3.6 Correlations

This DDoS is known since 1999. It was first known from the flooding at University of Minnesota which were originating from thousands of machines. There are a number of papers describing Trin00 attack in details such as SANS ID FAQ by Phillip Boyle, Trin00 Analysis by David Dittrich, University of Washington and the explanation of signatures from snort web site or white hat web site. (Please refer to reference for URL links.)

CAN-2000-0138 for information on a system has a distributed denial of service (DDOS) attack master, agent, or zombie installed, such as Trinoo.

2.3.7 Evidence of active targeting

The target in this case would be unknown victim machine. Host 10.10.10.165 is the Trin00 master and could be used to instruct the Daemon to launch the DDoS. This is not the target.

2.3.8 Severity

Severity = (criticality + lethality) - (system countermeasures + network countermeasures)

Criticality 1: The target is unknown in this case. Although Trin00 Master host is found on the network, no response from suspected Daemon. If the communication betweek Master and Daemon was successful, I would rate this higher.

Lethality 1: The attack is not successful.

System countermeasures 2: This risk can be mitigated by patching the system and reconfigure the security configuration such as password policy as poor password an improper config could also be used to install the handlers that will take part in the attack.

Network countermeasures 3: I do not have enough information on how the firewall is set up. Anyhow, the firewalls and routers could be used to filter out traffic with Trin00 relevant ports such as 27665, 27444 and 31335, and deny everything except port that is absolutely necessary.

Severity = (1 + 1) - (2 + 3) = -3

2.3.9 Defensive recommendation

To prevent internal hosts from taking part of the DDoS Trin00 attack or being used as master or daemon to launch the attack:

- Rebuild host 10.10.10.165.
- Keep all hosts up-to-date with latest patches and virus signatures.
- Disable unused network services.
- The firewall and router should be configured to filter out ports used by Trin00 such as 27665, 27444 and 31335.

- For Unix Server: use network access control tools such as TCP Wrappers to limit access to the internal network, use system integrity checks such as Tripwire to prevent rootkit from being installed.
- For Windows Server: scan the server with wintrinoo, a Trin00 scanning tool.

2.3.10 Multiple choice questions

What kind of attack involved the following ports 27665, 27444 and 31335?
A. Sub Seven
B. TFN
C. Trin00
D. Stacheldraht

Answer: C

2.3.11 References

[1] Distributed Denial of Service Attack Tools: trinoo and wintrinoo, URL: http://www.sans.org/resources/idfaq/trinoo.php
[2] Max Vision, Judy Novak. "Snort Signature Database". Snort Website. Snort \DDOS Trin00 master to daemon.htm URL: http://www.snort.org/snort-db/sid.html?sid=237
[3] David Dittrich. "The DoS Project's "trinoo" distributed denial of service attack tool". University of Washington. URL: http://www.secinf.net/uplarticle/1/trinoo_analysis.txt (October 1999)

Assignment 3 – Analyze This

3.1 Executive Summary

I have been assigned to perform the analysis of the university IDS logs captured by snort for a period of 5 days from 2004-04-07 to 2004-04-11. Three different types of files were downloaded. Altogether there are 15 log files including 5 alert files, 5 scan files and 5 out of spec files. In total, there are 90,844 alerts, 5449 out of spec packets and more than 15 million port scan packets! Some of the log entries were not properly formatted and cannot be used for analysis. Hence, I have ignored these entries and paid attention to those packets that were correctly formatted only. It is noted that OOS logs were not left out.Attention had been paid on well-formed packets that were correctly formatted.

This report provides a summary of critical alerts detected, top external hosts and top internal hosts that generated the alerts. Ten different types of alert base on number of occurrences were selected for analysis.

Summary

- There were various attack attempts against the university hosts. The attempts are from both internal hosts and external hosts. Some attack could seriously compromise the network.
- Some attacks could be successful and some hosts may have been compromised.
- A number internal host may have been infected by virus, worms or Trojan horses.
- Some internal hosts could be used as handler/agent to perform DDoS against other hosts.
- Some internal hosts have dangerous services running such as ftp, telnet, and etc.

Risks

- The Attacker may have already compromised some hosts.
- The Attacker may have plant Trojans for the purpose of returning to the host later.

Immediate actions required

- Investigate some hosts as they are likely to be infected by virus and worm or may have been compromised. (Immediately) Top 5 hosts are MY.NET.43.3, MY.NET.43.2, MY.NET.112.152, MY.NET.82.79, and MY.NET.84.235. (Please also refer to section 3.11 for a list of internal hosts that need to be investigated.)
- Update the virus signatures for all hosts.
- Turn off unneeded services.
- Update Snort to the latest version d fine tune snort rules to reduce the false alerts and noises so that the real attack can be clearly identified.
- Perform security patch and system hardening on every host.
- Reconfigure router and firewall, such as block some services, for better protection of the internal network.
- Enforce security policy such as personal firewall in all student's machines.

3.2 A list of files

The log files were downloaded from www.incidents.org and dated 2004-04-07 to 2004-04-11 inclusively. Following is a list of each log and its size.

Scan File	Size (MB)	Alert File	Size	OOS File	Size

			(MB)		**(MB)**
scans.040407	244.7	alert.040407	16.5	oos_report_040407	7.1
scans.040408	76.0	alert.040408	40.5	oos_report_040408	3.3
scans.040409	168.9	alert.040409	44.8	oos_report_040409	0.5
scans.040410	314.3	alert.040410	55.1	oos_report_040410	1.6
scans.040411	211.9	alert.040411	45.2	oos_report_040411	0.3

It is note that the packet date in the file were not align with the file name
Some files were incorrectly formatted and cannot be used for analysis so I ignored those packets. Following are examples of entries extracted from alert.040407 file.

```
04/07-15:12:38.344673  [**] EXPLOIT x86 NOOP [**] 199.131.21.3404/07-
15:21:16.861690  [**] SMB Name Wildcard [**] MY.NET.111.228:137 ->
209.2.144.10:137
:4041 -> MY.NET.84.204:80
```

or

```
04/07-18:54:35.390670  [**] EXPLOIT x86 NOOP [**] 67.50.96.24304/07-
19:27:14.017967  [**] spp_portscan: End of portscan from 69.9.244.52:
TOTAL time(0s) hosts(1) TCP(1) UDP(0) STEALTH [**]
:1143 -> MY.NET.83.67:80
```

Each of the above packets contains 2 alerts, which should have been broken down into 2 different entries. When converting the above alerts into MS access, only one would get converted and the other one would be shown as error. Manual effort is required to detect such mis-formatted packets so I have ignored the one that show error message.

3.3 Analysis methodology

I combined each types of file (alert files, scan files and oos files) into one using unix cat command. Then I used perl scripts developed by Terry MacDonald [11] to convert the file into csv format. (Thank you Terry.) After that the data were imported to Microsoft Access and MySQL database. I wrote a number of queries to summarize the data in different angles to help me understand the network architecture and be able to identify the role of each host and the relationship among the hosts. I sorted the data by number of occurrences. Activities that generated a lot of alerts were selected for analysis. The reason being the need to understand if these alerts are false alerts or are actual alerts that need to be investigated.

Traffic directions were analysed to understand how the vulnerabilities were exploited and how the university host took part in the attack such as the university host was a victim, a handler/agent for Trojan horse or the attacker itself. For those attacks that involved external hosts, I have checked with ARIN's WHOIS database.

I also checked the correlation data with the previous papers from various GIAC, please refer to the reference at the end of this paper.

3.4 Alert Log Analysis

Following is a list of all alerts generated during the period of 2004-04-07 to 2004-04-11. Altogether there were 90,845 alerts sorted by number of occurrences below.

Alerts	No. of alerts	Brief Description
EXPLOIT x86 NOOP	28072	The X86 NOP signature is triggered by continuous 0x90 characters. This could be an attempt to run attack via a buffer overflow exploit on X86 machine.
MY.NET.30.3 activity	12246	Activities with MY.NET.30.3.
SMB Name Wildcard	11803	There were attempts to scan port 137 for shared resources available. Some users may have shared the entire drive, rather than just the subdirectory.
High port 65535 tcp - possible Red Worm - traffic	10226	The alert show the possibility that some internal hosts may have been infected with Red Worm. The signature captures any packet with either the source port or the destination port is 65535. This worm is now known as Adore worm.
MY.NET.30.4 activity	10074	Activities with MY.NET.30.4.
Tiny Fragments - Possible Hostile Activity	8010	Packets with small fragmentation sent to the network to perform hostile activity. Usually this technique is used to evade the IDS.
DDOS mstream handler to client	3258	Msteam is a Distributed Denial of Service tool. Usually it works like this Attacker → Handler → Agent → Victim. This alert is generated when the handler is communicating with the compromised host or agent.
Possible trojan server activity	1057	This alerts show various possible Trojan activities via port 27374 such as Bad Blood, SubSeven, SubSeven Gold, and Subseven DefCon.
Null scan!	1032	This is a stealth scan to determine if the target host is alive. The attacker send TCP port scan with all flags turned off.
NMAP TCP ping!	1024	The attacker use nmap to probe the server to determine if the server is reachable.
External RPC call	930	RPC is a protocol that allows one program can use to request a service from another program located

Alerts	No. of alerts	Brief Description
		on another computer across the network. This alert is generated when external host is trying to initiate connection with internal hosts via RPC port 111.
SUNRPC highport access	629	This alert indicates the attempt to access port 32771, a port used by Sun OS to manage RPC services on a host.
Incomplete Packet Fragments Discarded	496	TCP packets were discarded due to the Incomplete header. This could be caused by fragmentation attacks or transmission error.
TCP SRC and DST outside network	289	Both the source address and the destination address are from external network.
High port 65535 udp - possible Red Worm - traffic	241	Possible Red Worm udp packet.
ICMP SRC and DST outside network	167	ICMP packet with both the source address and the destination address from external network.
[UMBC NIDS] Internal MiMail alert	154	MiMail is a worm that spreads by e-mail. This alert is triggered if suspected Mimail worm packet is detected.
[UMBC NIDS IRC Alert] IRC user /kill detected possible trojan	144	This alert indicates possible Trojan activities.
DDOS shaft client to handler	140	Shaft is another DDoS tool. This alerts indicate the communication between shaft handler and shaft master.
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected ..	108	Backdoor.Sdbot is a Backdoor Trojan Horse that allows the attacker to control a computer by using IRC. This alert is triggered for any the suspicious sdbot packets.
FTP passwd attempt	100	This alert is triggered when TCP packet from external is sent with flag ACK to port 21 on the local network. The packet has 'passwd' in the content.
TCP SMTP Source Port traffic	83	This alert is triggered with any packet with source port 25 and no ACK set.
IRC evil - running XDCC	70	XDCC is a server offering shared pirated softwares or movies to IRC users such as warez group. This alert indicates the attempt to run XDCC.
EXPLOIT x86 setuid 0	66	This alert indicates that system call was detected with the attempt to set user identity to 0.
SMB C access	55	The attempt to access the default administrative share C$. If allowed, the attacker can access the C: filesystem. This could be part of other exploits.
[UMBC NIDS] External MiMail alert	46	Mimail is a worm spreading via email using its own SMTP engine. It usually arrive in the network via email attachment, a ZIP file containing an HTML and a compressed Win32 EXE file.
connect to 515 from outside	46	Connection to port 515, a printer daemon port and could be DoS exploited.

Alerts	No. of alerts	Brief Description
scan (Externally-based)	43	Scan packets.
EXPLOIT x86 setgid 0	32	This alert indicates that system call was detected with the attempt to become a member of group root.
EXPLOIT x86 stealth noop	28	This alert the possible attempt to overflow the internal host.
[UMBC NIDS IRC Alert] Possible drone command detected.	25	This alert the attempt to issue malicious command via IRC channel.
RFB - Possible WinVNC - 010708-1	19	WinVNC is a VNC server that allow you to view Windows desktop from any viewer. WinVNC has a number of vulnerabilities that could be exploited.
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Reque...	17	XDCC refers to IRC bots running file sharing programs. XDCC bots serve one or more usually large files for download using the DCC protocol. Its use is widely understood to be a protocol extension by which illegal content, usually MP3s or warez, can be listed and, subsequently, downloaded [34].
NIMDA - Attempt to execute cmd from campus host	15	This is a possible NIMDA packet.
Attempted Sun RPC high port access	14	The alert indicates traffic generated from external hosts with an attempt to access RPC services.
TFTP - Internal UDP connection to external tftp server	14	Connection to external tftp server. Internal hosts initiated traffic to external host port 69.
SYN-FIN scan!	13	This alert indicates the attempt to scan the internal network with TCP packet with SF flags set.
FTP DoS ftpd globbing	11	This alert indicates that a remote attacker may be attempting to crash the ftpd server software. This could be caused by other vulnerabilities.
EXPLOIT NTPDX buffer overflow	10	The alert shows attempt to buffer overflow the ntpd network time daemon.
EXPLOIT x86 NOPS	8	Snort generates alerts when it finds packets with no Operation and lots of 0x90.
DDOS mstream client to handler	6	Mstream is a DDoS tool. This alerts indicate the communication between client and handler.
Probable NMAP fingerprint attempt	6	This event is generated when the nmap port scanner and reconnaissance tool is used against a host.
TFTP - External TCP connection to internal tftp server	4	External connection to tftp servers inside the university.
NETBIOS NT NULL session	3	Null session are used to list shares and users on NT machine.
[UMBC NIDS IRC Alert] K\:line'd user detected	2	UMBC is The University of Maryland, Baltimore County. This is the alert relevant to IRC channel.
[UMBC NIDS IRC Alert] User joining XDCC channel detecte...	2	XDCC is a feature of IRC to periodically list the files (usually 1-5 large files) in the channel (chat room) which it is hosting, for people to download. This alert detects user joining the XDCC

Alerts	No. of alerts	Brief Description
		channel.
PHF attempt	2	This is web application security attack. The attacker exploits a vulnerable CGI script to execute arbitrary commands.
Fragmentation Overflow Attack	1	This alert indicates the attempt to bring down the server.
External FTP to HelpDesk MY.NET.70.50	1	External ftp to internal host MY.NET.70.50.
External FTP to HelpDesk MY.NET.53.29	1	External ftp to internal host MY.NET.53.29.
External FTP to HelpDesk MY.NET.70.49	1	External ftp to internal host MY.NET.70.49.
Total	90844	

Below is the analysis of the selected events of interest. The main criteria for selecting the alerts are mixed between the number of occurrences and the criticality of the attack with the intention to cover wide variety of alert types such as Information Gathering or reconnaissance (covered in 3.4.2, 3.4.4, 3.4.9), Virus/Trojan/Worm alerts (covered in 3.4.3, 3.4.5, 3.4.11), Possible DoS (covered in 3.4.1), IRC attacks (covered in 3.4.6), RPC attacks (covered in 3.4.7) and FTP attacks (covered in 3.4.10).

3.4.1 EXPLOIT x86 NOOP

Description:
This alert accounts for 31% of all alerts generated for a period of 5 days. The alert indicates the attempt to run attack via buffer overflow exploit on X86 architecture machine by sending packets with contiguous bytes. Usually the signature will be triggered by a packet containing large piece of 0x90 characters.

Sample Alerts:
Sources of this alert are generated from external hosts. Below is a list of top talkers that generated the alert.

Alert	Destination IP	No. of alerts
EXPLOIT x86 NOOP	MY.NET.84.236	1056
EXPLOIT x86 NOOP	MY.NET.17.3	850
EXPLOIT x86 NOOP	MY.NET.70.74	832
EXPLOIT x86 NOOP	MY.NET.84.235	831
EXPLOIT x86 NOOP	MY.NET.84.204	448
EXPLOIT x86 NOOP	MY.NET.82.93	425
EXPLOIT x86 NOOP	MY.NET.17.4	366
EXPLOIT x86 NOOP	MY.NET.53.84	364
EXPLOIT x86 NOOP	MY.NET.32.139	357

The NOOP signature is a number of contiguous bytes that could be no-operation machine language codes for a particular architecture. NOOPs are often used to pad out TCP options. NOOP can also be used to buffer overflow the server, so this alert is indicating that it may have found an attempt to run attack code via a buffer overflow exploit on an x86 architecture machine. The false positive rate for this alert is high. There is no correlation with OOS packets.

The destination hosts of the alert have destination port 80 and the source ports are all high ports so it seems like connection from high port to http port. These events may be generated when binary data is being transferred from the source machines. I think that this alert has high probability to be false positives as per the analysis above.

Correlations:
Robert Graham [12] described the possibility that this attack could be false positive.
Terry MacDonald [11] also refer to this paper and raised that some image files contain the sort of hexcodes in them to trigger this rule. And that means often they will trigger when users are accessing websites, or just transferring files between computers.

Recommendations:
1. Upgrade snort to the latest version.
2. Use latest snort rule to reduce the false alerts.
3. Modify $SHELLCODE_PORTS variable to reduce NOOP false alerts. (var SHELLCODE_PORTS ![80])

3.4.2 SMB Name Wildcard

Description:
SMB is one of the most popular protocols allowing you to share disks, printers, files and etc. SMB Name Wildcard alert indicates the attempt to enumerate windows hosts to get a list of NeTBIOS names. This is a search for resources such as shares and usernames. The client does not identify NetBIOS name but in stead, it uses '*' wildcard to query the host for its NetBIOS table, something like NetBIOS name table probe. By accessing system name table, information that could be obtained include [31]:

1. The NetBIOS name of the server.
2. The Windows NT workgroup domain name.

3. Login names of users who are logged into the server.
4. The name of the administrator account if they are logged into the server.

Sample Alerts:

The alert files showed about 1 packet per few seconds and source and destination ports are 137 which are normal used in Windows operation. This indicates that the file sharing might be active on the target host [31].

Month	Day	Time	Alert	Source IP	Source Port	Destination IP	Destination Port
Apr	7	13:49:22.656588	SMB Name Wildcard	MY.NET.111.228	137	209.2.144.10	137
Apr	7	14:00:05.947851	SMB Name Wildcard	MY.NET.111.228	137	209.2.144.10	137
Apr	7	14:13:09.644466	SMB Name Wildcard	MY.NET.111.228	137	209.2.144.10	137
Apr	7	14:13:15.851752	SMB Name Wildcard	MY.NET.111.228	137	209.2.144.10	137
Apr	7	14:13:26.698588	SMB Name Wildcard	MY.NET.111.228	137	209.2.144.10	137
Apr	7	14:13:28.199437	SMB Name Wildcard	MY.NET.111.228	137	209.2.144.10	137
Apr	7	14:13:37.541799	SMB Name Wildcard	MY.NET.111.228	137	209.2.144.10	137

Month	Day	Time	Alert	Source IP	Source Port	Destination IP	Destination Port
Apr	7	14:13:46.957927	SMB Name Wildcard	MY.NET.111.228	137	209.2.144.10	137
Apr	7	14:14:00.824257	SMB Name Wildcard	MY.NET.111.228	137	209.2.144.10	137
Apr	7	14:14:14.752001	SMB Name Wildcard	MY.NET.111.228	137	209.2.144.10	137
Apr	7	14:14:19.469293	SMB Name Wildcard	MY.NET.111.228	137	209.2.144.10	137

Following is a list of top IP that generate the alert. All targets are external. This is the indication of internal hosts performing reconnaissance before the actual attack.

Alert	Source IP	Source Port	Destination IP	Destination Port	No. of alerts
SMB Name Wildcard	MY.NET.11.7	137	169.254.0.0	137	4996
SMB Name Wildcard	MY.NET.11.7	137	169.254.25.129	137	1738
SMB Name Wildcard	MY.NET.111.228	137	209.2.144.10	137	991
SMB Name Wildcard	MY.NET.11.6	137	169.254.0.0	137	518
SMB Name Wildcard	MY.NET.190.95	137	219.250.48.44	137	239
SMB Name Wildcard	MY.NET.5.34	137	199.239.137.216	137	150
SMB Name Wildcard	MY.NET.29.30	137	199.239.137.216	137	149
SMB Name Wildcard	MY.NET.190.93	137	150.208.201.50	137	59
SMB Name Wildcard	MY.NET.75.13	137	64.211.50.36	137	42

Correlations:
There is no correlation with OOS packets. The alert files showed about 1 packet per second and destination port is 137 which are almost consistent with the ArachNIDS database, IDS177 netbios-name-query [31].

Recommendations:
1. Use latest snort rule to reduce the false positives.
2. The security policy firewall should add the rule to filter out NetBIOS traffic over IP. Especially external access to NetBIOS services must be blocked. This could be done by filtering out UDP packet to port 137.
3. Block traffic to and from the reserved address e.g. 10.x.x.x, 127.x.x.x and. etc.
4. Investigate the router why reserved address packets are forwarded.
5. Investigate MY.NET.11.7 for sending packets to the reserved address whether it's mis-configuration or attacker trying to confuse the local address.

3.4.3 High port 65535 - possible Red Worm – traffic

This alert includes:

Alert	No. of alerts
High port 65535 tcp - possible Red Worm - traffic	10226
High port 65535 udp - possible Red Worm - traffic	241

Description:
Port 65535 is a high port and there is no service registered to this port except for Trojans. This alert indicates that some hosts in the university are infected with red worm/adore. The worm scan the network to look for vulnerable Linux hosts that could be exploited with well-known vulnerabilities such as LPRng, rpc-statd, wu-ftpd and BIND. Once the vulnerable host is identified, the worm will replace the system binary (ps) with a trojaned version. The icmp program listens for a specific ICMP packet and once it is received, it opens a backdoor on TCP port 65535 to the system [1].

Following is a top communication list.

Alert	Source IP	Source Port	Destination IP	Destination Port	No. of alerts
High port 65535 tcp - possible Red Worm - traffic	141.157.102.155	65535	MY.NET.60.16	22	2693
High port 65535 tcp - possible Red Worm - traffic	MY.NET.60.16	22	141.157.102.155	65535	2168

High port 65535 tcp - possible Red Worm - traffic	62.77.191.33	65535	MY.NET.153.83	1330	608
High port 65535 tcp - possible Red Worm - traffic	MY.NET.153.83	1330	62.77.191.33	65535	307
High port 65535 tcp - possible Red Worm - traffic	MY.NET.97.213	3645	69.193.86.240	65535	68
High port 65535 tcp - possible Red Worm - traffic	69.193.86.240	65535	MY.NET.97.213	3645	66

External host 141.157.102.155 and MY.NET.60.16 seems to be communicating to each other. Let's look at sample details. This looks like normal SSH traffic.

Month	Day	Time	Source IP	Source Port	Destination IP	Destination Port
Apr	8	23:45:06.617679	141.157.102.155	65535	MY.NET.60.16	22
Apr	8	23:45:06.631306	MY.NET.60.16	22	141.157.102.155	65535
Apr	8	23:45:08.446529	141.157.102.155	65535	MY.NET.60.16	22
Apr	8	23:45:08.461483	MY.NET.60.16	22	141.157.102.155	65535
Apr	8	23:45:08.653856	141.157.102.155	65535	MY.NET.60.16	22

Month	Day	Time	Source IP	Source Port	Destination IP	Destination Port
Apr	8	23:45:08.794017	141.157.102.155	65535	MY.NET.60.16	22
Apr	8	23:45:08.955410	141.157.102.155	65535	MY.NET.60.16	22
Apr	8	23:45:09.535802	141.157.102.155	65535	MY.NET.60.16	22
Apr	8	23:45:09.761137	141.157.102.155	65535	MY.NET.60.16	22
Apr	8	23:45:10.001812	MY.NET.60.16	22	141.157.102.155	65535
Apr	8	23:45:10.031616	MY.NET.60.16	22	141.157.102.155	65535
Apr	8	23:45:10.054782	141.157.102.155	65535	MY.NET.60.16	22
Apr	8	23:45:10.091633	MY.NET.60.16	22	141.157.102.155	65535
Apr	8	23:45:10.221621	MY.NET.60.16	22	141.157.102.155	65535
Apr	8	23:45:10.245870	141.157.102.155	65535	MY.NET.60.16	22
Apr	9	01:00:46.752106	141.157.102.155	65535	MY.NET.60.16	22
Apr	9	01:00:46.771739	MY.NET.60.16	22	141.157.102.155	65535
Apr	9	01:00:46.965935	141.157.102.155	65535	MY.NET.60.16	22
Apr	9	01:00:48.812966	141.157.102.155	65535	MY.NET.60.16	22
Apr	9	01:00:48.851857	MY.NET.60.16	22	141.157.102.155	65535
Apr	9	01:00:48.911916	141.157.102.155	65535	MY.NET.60.16	22

The traffic show that 2 hosts are talking for a while. MY.NET.60.16 seems to be a SSH server and it's talking to external host 141.157.102.155 (Verizon, probably some dial up student) I have checked the OOS and scan file for activities performed by MY.NET.60.16 but nothing seems to be abnormal. I have also checked log files and look for internal correlation. There is nothing from traffic with port 22, this is false positive.

Correlations:

Carlin Carpenter's paper [24] and Les Gordon's paper [33]. The adore worm on udp traffic could be false alerts.

Recommendations:

Although the alert is false positive, I would strongly recommended the following recommendations.

1. Run adorefind on the suspicious hosts. Adorefind is a utility developed by Dartmouth's ISTS to detect hosts infected with Adore worm. The file can be downloaded at http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm.
2. For all linux hosts, keep the system up-to-date with latest patch from the vendors.
3. Block outgoing e-mail to the four e-mails addresses. (adore9000@21cn.com, adore9000@sina.com, adore9001@21cn.com, and adore 9001@sina.com), and block access to the go.163.com domain. [25]
4. Configure email server to block or remove email that contains file attachments that are commonly used to spread viruses, such as .vbs, .bat, .exe, .pif and .scr files. [26]

3.4.4 Tiny Fragments - Possible Hostile Activity

Description:

The tiny fragment refers to a small fragmentation packet that is sent to the network. It is usually used for hostile activity with the purpose to evade the IDS and the firewall that does not perform packet assembly. The first fragment is so small that it does not have IP header such as source port and destination port. The reassembly will take place at the destination host so this technique is often used to hide malicious activities.

From the detect below, the tiny fragment alerts were generated from host outside the network and clearly MY.NET.43.3 and MY.NET.112.218 are the target. From the analysis, MY.NET.43.3 is a web server with various services running. The attackers probably want to DoS this server.

Snort signature that generates this would be:

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC Tiny
Fragments"; fragbits:M; dsize: < 25; classtype:bad-unknown;
sid:522; rev:1;)
```

Any external host trying to initiate connection with internal hosts via any port, with fragmentation and reserved bits set in the IP header and packet payload size <25 bytes.

Sample Alerts:

Sample alerts are shown below.

M	Day	Time	Source IP	Source Port	Destination IP	Destination Port
Apr	10	02:40:01.174095	212.76.225.24	None	MY.NET.43.3	None
Apr	10	02:40:01.321031	212.76.225.24	None	MY.NET.43.3	None
Apr	10	02:40:01.335691	212.76.225.24	None	MY.NET.43.3	None
Apr	10	02:40:01.365705	212.76.225.24	None	MY.NET.43.3	None
Apr	10	02:40:01.392780	212.76.225.24	None	MY.NET.43.3	None
Apr	10	02:40:01.407511	212.76.225.24	None	MY.NET.43.3	None
Apr	10	02:40:01.440482	212.76.225.24	None	MY.NET.43.3	None

Top generators of this alert are listed below. I ran a check to see who are these external hosts that cause the IDS to alert. The information is listed on the second column.

Source IP	External host	Source Port	Destination IP	Destination Port	No. of alerts
212.76.225.24	Coditel - Internet Services in Belgium	None	MY.NET.43.3	None	7487
200.221.134.63	Comite Gestor – Internet, Brazil	None	MY.NET.112.218	None	263
200.221.134.147	Comite Gestor – Internet, Brazil	None	MY.NET.80.5	None	195
61.216.77.99	CHTD, Chunghwa Telecom Co.,Ltd. - Taiwan	None	MY.NET.12.6	None	20
212.76.225.24	Coditel - Internet Services in Belgium	None	MY.NET.43.2	None	15
24.93.213.53	Road Runner - ISP in the US	None	MY.NET.97.39	None	5
61.19.223.227	Communication Authority of Thailand – Internet gateway in Thailand	None	MY.NET.97.190	None	4

After analysed the OOS file for correlations, I find that these Tiny Fragments packets assembled into the XMAS or FULLXMAS scan, refer to the packet below:

```
04/10  02:40:01       212.76.225.24  19089   130.85.43.3    29333  FULLXMAS       *2UAPRSF
```

MY.NET.43.3 seems to be the target for tiny fragment attack. Investigation should be done to look for sign of compromises in this server.

Correlations:
Terry MacDonald paper [11].

Recommendations:

1. In a router, this can by prevented by enforcing certain limits on fragments passing through, namely, that the first fragment be large enough to contain all the necessary header information. [13]
2. Apply timeout for the fragmented packet.
3. Perform detail security assessment on MY.NET.43.3. There is a possibility that this host might be compromised.
4. Perform system hardening on MY.NET.43.3.

3.4.5 Possible Trojan Server Activity

Description:
This alert is triggered when there's possible Trojan activity detected in the network. From the alerts details, all packets that caused this alert has port 27374 as either the source port or the destination port. [14] [15] Port 27374 is being used by a number of Trojan horses such as Bad Blood, Fake SubSeven, li0n, Ramen, Seeker, SubSeven , SubSeven 2.1 Gold, Subseven 2.1.4 DefCon 8, SubSeven 2.2, SubSeven Muie, and The Saint. The famous one would be Sub Seven, a well-known remote access Trojan (RAT). This Trojan allows an attacker to control the compromised host completely and perform almost any activities remotely.

Port 27374 has been discovered as the listener port for Ramen.Linux worm which target some Red Hat hosts.[16] The worm then starts an HTTP server on port 27374 to serve out itself to newly infected machines and also patches the exploits that it used to gain access to the system.

Sample Alerts:
From the analysis, the top source IP addresses that generate these alerts are from external network.

Source IP	Source IP	No. of alerts
213.189.89.109	QualityNet Kwait - Internet Gateway in Kuwait	427
213.189.89.54	QualityNet Kwait - Internet Gateway in Kuwait	271

These 2 source IP perform internal host scanning to look for possible machine infected with SubSeven. The alerts below show that the scanning was done within seconds.

Month	Day	Time	Source IP	Source Port	Destination	Destination Port
Apr	10	16:08:29.350173	213.189.89.109	2619	MY.NET.190.59	27374

Apr	10	16:08:29.350080	213.189.89.109	2600	MY.NET.190.40	27374
Apr	10	16:08:29.350092	213.189.89.109	2606	MY.NET.190.46	27374
Apr	10	16:08:29.350105	213.189.89.109	2603	MY.NET.190.43	27374
Apr	10	16:08:29.350118	213.189.89.109	2617	MY.NET.190.57	27374
Apr	10	16:08:29.350133	213.189.89.109	2607	MY.NET.190.47	27374
Apr	10	16:08:29.350276	213.189.89.109	2618	MY.NET.190.58	27374
Apr	10	16:08:29.350159	213.189.89.109	2601	MY.NET.190.41	27374
Apr	10	16:08:29.349905	213.189.89.109	2609	MY.NET.190.49	27374
Apr	10	16:08:29.350188	213.189.89.109	2616	MY.NET.190.56	27374
Apr	10	16:08:29.350203	213.189.89.109	2605	MY.NET.190.45	27374
Apr	10	16:08:29.350219	213.189.89.109	2611	MY.NET.190.51	27374
Apr	10	16:08:29.350233	213.189.89.109	2608	MY.NET.190.48	27374
Apr	10	16:08:29.350247	213.189.89.109	2614	MY.NET.190.54	27374
Apr	10	16:08:26.423697	213.189.89.109	2606	MY.NET.190.46	27374
Apr	10	16:08:29.350146	213.189.89.109	2604	MY.NET.190.44	27374

I noticed that some of the MY.NET hosts did response to the traffic originating from outside network on source port 27374. This could mean that these hosts might have been infected with Sub Seven already. Look at sample alerts below:

Source IP	Source Port	Destination IP	Destination Port	No. of alerts
68.55.195.232	27374	MY.NET.12.6	25	33
MY.NET.12.6	25	68.55.195.232	27374	24

Source IP	Source Port	Destination IP	Destination Port	No. of alerts
MY.NET.24.44	80	170.91.5.4	27374	16
170.91.5.4	27374	MY.NET.24.44	80	14

Source IP	Source Port	Destination IP	Destination Port	No. of alerts
24.35.92.178	27374	MY.NET.24.34	80	8
MY.NET.24.34	80	24.35.92.178	27374	7

Although this could be false positive as port 27374 could be used as ephemeral port, I do not have enough information to confirm such possibility.

A list of internal hosts that responded to port 27374 is given in section 3.11.3.

Correlations:
Les Gordon's GCIA paper [33] analysed this alert and thought that port 27374 could be just ephemeral port and this alert could be false positive.

Recommendations:

1. Investigate suspicious internal hosts that might be infected with Trojans. (Please refer to 3.11.3.)
2. Turn off unneeded services at all hosts.
3. Keep patches up-to-date for all hosts.
4. Check if anti-virus software is installed on all hosts at the university and if virus signatures are up-to-date.

3.4.6 IRC Alerts

This is a set of alert messages related to IRC detected during the 5 days period.
IRC Evil – running XDCC
[UMBC NIDS IRC Alert] IRC user /kill detected possible trojan.
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot
[UMBC NIDS IRC Alert] K\:line'd user detected possible trojan
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC

Description:
IRC is a popular communication channel among the universities students and the internet surfers. However, it is not being used as a channel to chat only, but also a channel to send commands to the bots (automated malicious programs) on the compromised machine. With this channel, the attacker can launch various kinds of attack such as distributed denial of service or stealing information. Below is a number of alerts relevant to IRC that were generated from the university's IDS:

Alerts	No. of alerts
[UMBC NIDS IRC Alert] IRC user /kill detected possible trojan.	144
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	108
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	17
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	2
[UMBC NIDS IRC Alert] K\:line'd user detected possible trojan	2
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	1

IRC user /kill detected possible Trojan – These alerts are generated from external hosts using various Trojan ports, see a list below. IRC servers usually accept connections on ports 6660 to 6669 and sometimes port 7000 as well. Although these ports are commonly used, there are a number of Trojans using these ports as well. Port 7000 and 6667 could be Aladino, Gunsan, Remote Grab, SubSeven, SubSeven 2.1 Gold, Theef. Port 6669 is known for "Voyager Alpha Force" Distributed Denial of Service Agent. The infected machines will use outgoing 6669 port to connect to the IRC in order to call a bot to scan for other MS-SQL and MSDE machines using TCP port 1433, and to launch denial of service (DoS) attacks. When installed and run the IRC Bots usually try to connect to IRC ports [21].

Sample Alerts:
From the analysis, source IP are from external hosts with the following source ports:

Alerts	Source Port	No. of alerts
[UMBC NIDS IRC Alert] IRC user /kill detected possible trojan.	7000	73
[UMBC NIDS IRC Alert] IRC user /kill detected possible trojan.	6667	45
[UMBC NIDS IRC Alert] IRC user /kill detected possible trojan.	6669	14
[UMBC NIDS IRC Alert] IRC user /kill detected possible trojan.	6666	5

Considering timing between each alert, these could be just IRC connection between host 128.122.66.204 (New York University) and MY.NET.112.152.

Month	Day	Time	Source IP	Source Port	Destination IP	Destination Port
Apr	7	16:32:44.188487	128.122.66.204	7000	MY.NET.112.152	1126
Apr	7	16:49:29.044689	128.122.66.204	7000	MY.NET.112.152	1042
Apr	7	16:50:01.069207	128.122.66.204	7000	MY.NET.112.152	1043
Apr	7	16:50:35.232153	128.122.66.204	7000	MY.NET.112.152	1045
Apr	7	16:51:07.023492	128.122.66.204	7000	MY.NET.112.152	1052
Apr	7	16:52:22.020376	128.122.66.204	7000	MY.NET.112.152	1060
Apr	7	16:53:26.048966	128.122.66.204	7000	MY.NET.112.152	1066
Apr	7	16:56:20.370960	128.122.66.204	7000	MY.NET.112.152	1079
Apr	7	16:56:53.036192	128.122.66.204	7000	MY.NET.112.152	1080
Apr	7	16:58:30.026258	128.122.66.204	7000	MY.NET.112.152	1085
Apr	7	16:59:02.041878	128.122.66.204	7000	MY.NET.112.152	1086
Apr	7	17:00:06.005649	128.122.66.204	7000	MY.NET.112.152	1092
Apr	7	17:00:48.997883	128.122.66.204	7000	MY.NET.112.152	1094

K\:line'd user detected possible Trojan – External hosts sent packet to internal hosts via Trojan port (6883 and 6969)

Source IP	Source Port	Destination IP	Destination Port	No. of alerts
211.146.117.228	6883	MY.NET.84.203	1181	1
210.155.158.200	6969	MY.NET.97.158	3416	1

Possible sdbot floodnet detected attempting to IRC – This is another Trojan program that could flood the target machine From the alerts, a number of MY.NET hosts were trying to connect to the following destination IP via port 7000. I suspected these machines could be infected with sdbot Trojan and could be the agents to connect to the IRC servers at various university (The Handler).

Destination IP	Destination Port	No. of alerts
128.122.66.204 New York University	7000	104
131.96.118.15 Georgia State University	7000	2
146.151.53.178 University of Wisconsin	7000	1
141.64.6.71	7000	1

RIPE Network Coordination Centre		

IRC Evil – running XDCC, Possible Incoming XDCC Send Request Detected – XDCC allows file sharing through IRC channels. I will analyse these 2 alerts together as they looked relevant. Most of the 'IRC evil – running XDCC' alerts are generated by the internal hosts MY.NET43.2 and MY.NET.82.79.

Month	Day	Time	Source IP	Source Port	Destination IP	Destination Port
Apr	8	02:42:27.157667	MY.NET.43.2	1916	64.246.60.72	6667
Apr	8	02:44:58.019310	MY.NET.43.2	1916	64.246.60.72	6667
Apr	8	02:46:50.010611	MY.NET.43.2	1916	64.246.60.72	6667
Apr	8	02:48:56.045980	MY.NET.43.2	1916	64.246.60.72	6667
Apr	8	02:52:38.182079	MY.NET.43.2	1916	64.246.60.72	6667

It seems that the internal hosts were trying to create connection with external IRC host. Top destination hosts are 64.246.60.72 (Everyones Internet, Inc.) and 207.36.180.241 (CyberGate, Inc.) via port 6667 and port 6663 respectively.

From the analysis, the infected MY.NET hosts connected to the IRC server and the IRC Server sent back some command, which could be a legitimate command or a malicious one. There is not enough information to find out if these hosts have been compromised.

Alerts	Month	Day	Time	Source IP	Source Port	Destination IP	Destination Port
IRC evil - running XDCC	Apr	8	02:52:38.182079	MY.NET.43.2	1916	64.246.60.72	6667
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	Apr	8	03:00:11.011582	64.246.60.72	6667	MY.NET.43.2	1916

Alerts	Month	Day	Time	Source IP	Source Port	Destination IP	Destination Port
IRC evil - running XDCC	Apr	11	02:14:28.455067	MY.NET.82.79	1275	207.36.180.241	6663
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	Apr	11	02:42:03.848104	207.36.180.241	6663	MY.NET.82.79	1275

Correlations:
There's a number of articles on how to use IRC to launch the attack especially the DDOS attack. It is possible that some IRC users could be used as the agent and there is no information to conclude if the attack is successful. Terry MacDonald's GCIA paper has an analysis on the university hosts infected with the virus.

Recommendations:
1. Investigate MY.NET.43.2, MY.NET.112.152 and MY.NET.82.79 immediately.
2. All IRC users should have personal firewall and latest virus signature installed.
3. Patch all IRC servers at the university.

3.4.7 RPC Alerts

This is a set of alert messages related to RPC including:

Alert	No. of alerts
External RPC call	930
SUNRPC highport access!	629
Attempted Sun RPC high port access	14

Description:
RPC is a protocol on UNIX developed to facilitate one program in calling the service in another program across the network, without knowing the network details. There are a number of vulnerabilities related to RPC which could allow the attacker to perform various malicious activities ranging from buffer overflow to gain unauthorized access to the system. To exploit this, the attack will need to scan the target host if RPC service is running. Then, find out specific RPC services and port the service is running on. Usually the attacker will scan port 111 which is being used by portmapper, a program that manage RPC services on a host, or scan port 32771, some sun OS listens at this port for portmapper. The attacker can directly scan the ports range to see the services running.

Sample Alerts:
The alert messages were triggered by the external hosts trying to create connection to internal hosts through the following ports.

Alerts	Destination Port
External RPC call	111
SUNRPC highport access!	32771
Attempted Sun RPC high port access	32771

We discovered that the following hosts have scanned a number of internal hosts to see if RPC service is running. The following are 2 external machines that perform the RPC port scanning through port 111.

Alert	Source IP	No. of alerts
External RPC call	213.46.246.46 RIPE Network Coordination Centre	670
External RPC call	217.160.94.163 RIPE Network Coordination Centre	260

217.160.94.163 scanned MY.NET.5.5, MY.NET.6.15, MY.NET.6.15, MY.NET.16.90, MY.NET.16.106 and MY.NET.16.114, MY.NET.190.1 - MY.NET.190.254 within few minutes.

Usually the scanning packets will have the same source ip but different source port and different destination ip. But I noticed there are some odd scanning packets from 213.46.246.46 which used the same source port to scan the destination ip more than once.

Month	Day	Time	Alert	Source IP	Source Port	Destination IP	Destination Port
Apr	10	10:26:44.020320	External RPC call	213.46.246.46	46199	MY.NET.5.5	111
Apr	10	10:26:47.385070	External RPC call	213.46.246.46	46199	MY.NET.5.5	111
Apr	10	10:26:48.025597	External RPC call	213.46.246.46	46199	MY.NET.5.5	111

Month	Day	Time	Alert	Source IP	Source Port	Destination IP	Destination Port
Apr	10	10:26:48.200047	External RPC call	213.46.246.46	46465	MY.NET.6.15	111
Apr	10	10:26:48.313295	External RPC call	213.46.246.46	46465	MY.NET.6.15	111
Apr	10	10:26:48.318236	External RPC call	213.46.246.46	665	MY.NET.6.15	111
Apr	10	10:26:48.431399	External RPC call	213.46.246.46	665	MY.NET.6.15	111
Apr	10	10:26:48.545415	External RPC call	213.46.246.46	665	MY.NET.6.15	111
Apr	10	10:26:48.546179	External RPC call	213.46.246.46	665	MY.NET.6.15	111
Apr	10	10:26:48.659000	External RPC call	213.46.246.46	665	MY.NET.6.15	111

I suspected that the alerts generated were caused by the attempt to scan if RPC service was running on the host so the attacker can perform the actual attack on the server. We could not find if the above destination host were compromised with the highport access. This could be just the information gathering.

All 'SUNRPC highport access' alerts are generated form external hosts. For example, external host 64.12.25.40 tried to access MY.NET.82.106 with port 32771.

Month	Day	Time	Source IP	Source Port	Destination IP	Destination Port
Apr	8	13:47:13.310493	64.12.25.40	5190	MY.NET.82.106	32771
Apr	8	13:47:13.330167	64.12.25.40	5190	MY.NET.82.106	32771
Apr	8	13:47:13.330297	64.12.25.40	5190	MY.NET.82.106	32771
Apr	8	13:47:13.330399	64.12.25.40	5190	MY.NET.82.106	32771
Apr	8	13:47:13.333600	64.12.25.40	5190	MY.NET.82.106	32771
Apr	8	13:47:13.333721	64.12.25.40	5190	MY.NET.82.106	32771
Apr	8	13:47:13.512514	64.12.25.40	5190	MY.NET.82.106	32771

Apr	8	13:47:14.206591	64.12.25.40	5190	MY.NET.82.106	32771

Some MY.NET. hosts should be investigated immediately as there were alert messages that RPC high port has already been accessed. (Please refer to 3.11.2)

Correlations:
Erik Montcalm's GCIA paper [22] analysed that the External RPC call could be the reconnaissance.

Recommendations:
1. Block port 111 at the firewall. This might not entirely solve the problem but it will reduce the possibility of scanning and a number of alerts.
2. Also block all other RPC ports 32771-34000 at the firewall.
3. Maintain up-to-date patches on the hosts.
4. Investigate MY.NET.6.15 and MY.NET.5.5 if RPC is running on these hosts.

3.4.8 My.NET.30.3 activity, My.NET.30.4 activity

These 2 alerts account for 24.5% of the total alert messages. The alert message does not give any information that will be useful on the suspicious activity and I do not have information on the signature that generated this alert.

Alert	No. of alerts
MY.NET.30.3 activity	12246
MY.NET.30.4 activity	10074

Description:
From the analysis I think that these 2 hosts could be the web servers. All alert messages generated from external hosts indicating attempts to connect to these 2 servers with the following ports. I put the service name registered to port in the bracket.

Sample Alerts:
Sample alert is shown below.

Month	Day	Time	Source IP	Source Port	Destination IP	Destination Port
Apr	10	15:19:50.518740	68.55.113.194	32949	MY.NET.30.3	524
Apr	10	15:53:42.824190	68.55.27.157	1035	MY.NET.30.3	524
Apr	10	15:19:52.306355	68.55.113.194	32949	MY.NET.30.3	524
Apr	10	15:19:52.281265	68.55.113.194	32949	MY.NET.30.3	524

Top destination ports used by the external hosts are as follows:

Destination IP	Destination Port	No. of alerts

MY.NET.30.3	524 (Netware Core Protocol)	11568
MY.NET.30.3	80 (Web Server)	439
MY.NET.30.3	2745 (Bagle Virus Backdoor)	72
MY.NET.30.3	6129 (Dameware remote admin)	42
MY.NET.30.3	4899 (Remote Administrator default port)	20

Destination IP	Destination Port	No. of alerts
MY.NET.30.4	51443 (Novell Secure Folder)	7255
MY.NET.30.4	80 (Web Server)	2140
MY.NET.30.4	524 (Netware Core Protocol)	442
MY.NET.30.4	2745 (Bagle Virus Backdoor)	74
MY.NET.30.4	6129 (Dameware remote admin)	57
MY.NET.30.4	4899 (Remote Administrator default port)	20

Port 524 is used by Novell for communication between Novell clients and servers and for time synchronization between IP servers and supposed to be used internally. Port 80 is web server. Some other destination ports are Trojans such as 2745, 6129, and 4899. These 2 hosts might be web server. Anyway, I do not have enough information to do further analysis on this.

Correlations:
Erik Montcalm [22] analysed these 2 alerts in his GCIA paper but the information is limit as well. From his paper, a lot of external hosts also tried to connect to MY.NET.30.4 via 80, 524 and 51443.

Recommendations:

1. Perform detail security assessment against these 2 hosts to detect what are other vulnerabilities exist in the system. And perform system hardening.
2. Install latest patch.
3. This server should be placed in the right zoning in the university's network infrastructure and proper security measures implemented. If this is a web server, firewall should be configured to block all unnecessary ports such as all the above ports except for port 80 or 443.

3.4.9 Scan, Probe

This is a set of alert messages generated by the activities I believed to be the reconnaissance.

Alerts	No. of alerts
Null scan!	1032
SYN-FIN scan!	13
NMAP TCP Ping	1024
Probable NMAP fingerprint attempt	6

Description:
Before performing any penetration, the attacker usually perform basic information gathering step by the use of various types of scan. The results of the scan will provide necessary information on the target such as the existence of hosts and ports running. This information will be used to determine the type of attack to be performed against the target hosts. From the alert messages, some type of reconnaissance that were performed against MY.NET network include:

Null scan – A scanner send packet without TCP flag to the destination. This scan is used to determine the operating system of target hosts. This is most likely to be performed by NMAP.
NMAP TCP Ping – External hosts used NMAP to send ICMP packet to internal host to determine if hosts are alive. The command to be used is nmap –sS –PT –O <ip>.
SYN-FIN scan – External hosts sent packets with SYN and FIN flags enabled to internal hosts. This technique is used to determine if host is alive or fingerprint the operating system.
Probable NMAP fingerprint attempt – External hosts used NMAP to determine operating system being run on the target hosts.

Sample Alerts: Sources are from external hosts, indicating various attempts to gather information in order to attack the university's network. Sample alerts are as follow:

Alert	Month	Day	Source IP	Source Port	Dest IP	Dest Port
Null scan!	Apr	11	219.137.39.207	11423	MY.NET.84.235	54099
Null scan!	Apr	11	219.137.39.207	53545	MY.NET.84.235	113
Null scan!	Apr	11	219.137.39.207	None	MY.NET.84.235	None
Null scan!	Apr	7	82.64.84.58	None	MY.NET.42.3	None

Some of the null scan packets also appeared in the OOS packets. For the null scan packets without source port and destination port are very unlikely as null scan can only occur with TCP and this is not the fragmented packet as the alerts must contain TCP header (flags, ack, seq) So I think this could be a bug.

I find that some of the source IP also appeared in OOS packet as XMAS scan.

```
04/07 16:41:28    82.64.84.58 0     130.85.42.3 0     XMAS  **U*P**F
```

This means that the above host was trying various scans to gather information on the destination host.

lert	Month	Day	Time	Source IP	Source Port	Dest IP	Dest Port
SYN-FIN scan!	Apr	10	13:58:37.791911	198.92.146.22	80	MY.NET.97.136	1155
SYN-FIN scan!	Apr	10	02:43:40.384030	138.23.236.133	3692	MY.NET.24.47	1068

SYN-FIN scan!	Apr	8	17:30:13.766324	63.208.234.245	8080	MY.NET.97.54	1097
SYN-FIN scan!	Apr	8	16:20:23.950790	4.14.37.189	6881	MY.NET.43.3	2435

Alert	Month	Day	Time	Source IP	Source Port	Dest IP	Dest Port
NMAP TCP ping!	Apr	7	14:28:26.780416	12.158.155.194	80	MY.NET.1.4	53
NMAP TCP ping!	Apr	7	14:28:26.889825	65.241.119.130	80	MY.NET.1.4	53
NMAP TCP ping!	Apr	7	14:56:53.108457	216.239.183.2	81	MY.NET.24.44	80
NMAP TCP ping!	Apr	7	15:00:43.394883	63.211.17.228	80	MY.NET.1.3	53

Alert	Month	Day	Time	Source IP	Source Port	Dest IP	Dest Port
Probable NMAP fingerprint attempt	Apr	10	21:42:23.730396	63.211.210.20	80	MY.NET.97.18	1483
Probable NMAP fingerprint attempt	Apr	10	19:41:04.308981	4.47.65.115	3085	MY.NET.24.44	80
Probable NMAP fingerprint attempt	Apr	10	16:23:48.645396	200.63.130.10	23113	MY.NET.34.14	13171

Top target IPs for the reconnaissance are:

Destination IP	No. of alerts
MY.NET.1.3	548
MY.NET.12.6	196
MY.NET.12.4	177
MY.NET.111.34	73
MY.NET.111.34	72
MY.NET.1.4	70
MY.NET.82.79	65
MY.NET.43.3	57
MY.NET.153.35	51

Correlations:
Most of the previous papers have analysis on different types of scans from the out-of-spec files. Scan is used for the reconnaissance purpose.

Recommendations:
1. Check the top destination hosts for sign of compromises.
2. Secure all internal hosts by installing latest patch and perform system hardening.
3. Utilize stateful firewall and set up the configuration properly to block

these attempts.

4. Review router's configuration set up. Router should be configured to block simple port scan.

3.4.10 FTP Alerts

This include the following alerts:

Alerts	Destination IP	No. of alerts
FTP passwd attempt	MY.NET.24.47	100
FTP DoS ftpd globbing	MY.NET.24.27	11

Description:
FTP is the file transfer protocol. There are numerous attacks which can be done against ftp servers. In this environment, I understand that MY.NET.24.47 and MY.NET.24.27 are ftp servers.

Globbing is a feature in some FTP product that allows user to do a path name search. The attacker could create denial of service by sending the wildcard request to vulnerable ftp hosts [27]. The IDS will look for |2f2a| in the packet, for example */../*/../ or */.*/*/ . This could be false positive if the source machine makes a legitimate wildcard request.

Sample Alerts:

Alert	Month	Day	Time	Source IP	Source Port	Destination IP	Dest Port
FTP passwd attempt	Apr	8	01:56:46.851562	202.20.73.30	49561	MY.NET.24.47	21
FTP passwd attempt	Apr	8	01:56:47.080368	202.20.73.30	49561	MY.NET.24.47	21
FTP passwd attempt	Apr	8	01:56:47.538779	202.20.73.30	49561	MY.NET.24.47	21
FTP DoS ftpd globbing	Apr	10	11:53:58.127486	140.239.150.248	3387	MY.NET.24.27	21
FTP DoS ftpd globbing	Apr	10	11:53:59.239259	140.239.150.248	3387	MY.NET.24.27	21
FTP DoS ftpd globbing	Apr	10	11:54:00.367257	140.239.150.248	3387	MY.NET.24.27	21
FTP DoS ftpd globbing	Apr	10	11:54:01.611847	140.239.150.248	3387	MY.NET.24.27	21
FTP DoS ftpd globbing	Apr	10	12:01:51.020061	140.239.150.248	3387	MY.NET.24.27	21

Various external hosts have tried to connect to these hosts and some might have succeeded compromise the host. Although I do not have enough information to conclude this as we do not have response to the stimulus, there is a high possibility that such attack happened as the ftp data stream usually not contain "passwd" or wildcard request.

Correlations:
I could not find the analysis on previous assignments relevant to this.

Recommendations:
1. Investigate MY.NET.24.27 and MY.NET.24.47 immediately.
2. Patch the server with latest file from the operating system vendor.
3. Secure FTP server by disable anonymous access, enable log, configure ACL, user logon time restriction, and set up strong password policy.

3.4.11 NIMDA - Attempt to execute cmd from campus host

Description:
NIMDA is a worm targeting windows machine. The infected machine will try to transfer a nimda code to the vulnerable IIS server via tftp which can lead to DoS. Following is the life cycle of Nimda worm.
- Nimda locates EXE files from local machine
- Nimda locates e-mail addresses as well as searching local HTML files for additional addresses
- Nimda scan the Internet, locate www servers. Once found, Nimda will modify web pages.
- Nimda search for file shares in the local network.

The alert packets below show the internal hosts that might be infected with NIMDA connect to the external web servers. The packets below could have classified as normal http traffic if not because of the cmd.exe found in the data stream. The best correlation for this case is to obtain the web server log and analyse to check that this not false alert. However, such information is not available.

Month	Day	Time	Source IP	Source Port	Destination IP	Destination Port
Apr	8	05:29:42.925507	MY.NET.97.36	3670	69.90.32.141	80
Apr	8	07:20:44.479101	MY.NET.10.79	1091	64.70.33.115	80
Apr	8	17:30:26.611199	MY.NET.97.228	3163	69.90.32.141	80
Apr	8	17:30:56.244930	MY.NET.97.228	3174	69.90.32.141	80
Apr	8	17:30:56.613572	MY.NET.97.228	3174	69.90.32.141	80
Apr	8	23:16:33.750945	MY.NET.97.166	1269	69.90.32.141	80
Apr	8	23:16:34.534091	MY.NET.97.166	1269	69.90.32.141	80
Apr	9	16:55:44.014937	MY.NET.97.69	3285	69.90.32.141	80

Apr	10	11:09:31.015313	MY.NET.97.180	1593	216.64.193.20	80
Apr	10	19:53:18.715368	MY.NET.97.74	4213	69.90.32.141	80
Apr	10	19:53:20.493171	MY.NET.97.74	4213	69.90.32.141	80

Correlations:
Richard Baker provided details explanation on Nimda worm in his GCIA paper. [29]

Recommendations:
1. Investigate the infected hosts immediately.
2. Apply ingress and egress content filtering to filter out the nimda traffic from entering the university's network.
3. Patch all internal Windows hosts.
4. Update virus signatures.
5. Educate users not to open suspicious mails.

3.5 Scan Log Analysis

The 5 scan files make one extremely large combined file with the size of almost 1 GB. I used the script to convert the file into csv and import the data to mysql database. The total number of scan records is 15,667,034! Some of the records were incorrectly format and cannot be used for analysis so I removed all those records. The number of records by scan type is shown in the table below.

Scan type	Count	%
SYN	9216423	58.8273%
UDP	6389305	40.7822%
FIN	58798	0.3753%
INVALIDACK	823	0.0053%
NULL	504	0.0032%
NOACK	490	0.0031%
UNKNOWN	316	0.0020%
VECNA	169	0.0011%
XMAS	26	0.0002%
FULLXMAS	22	0.0001%
SYNFIN	15	0.0001%
SPAU	12	0.0001%
NMAPID	8	0.0001%
SYApr	2	0.0000%
SYNApr	2	0.0000%
Total	15666915	100%

SYN and UDP scans account for 98% of total scan. The following are some statistics from the scan file.

Top Source IP - Internal

Source	No. of Scan
130.85.111.51	1621815
130.85.153.35	1522455
130.85.81.39	1187999
130.85.70.96	1130696
130.85.112.152	1082053
130.85.1.4	795875
130.85.66.56	334888
130.85.84.235	294843
130.85.42.2	253164
130.85.53.169	237464
130.85.34.14	233328
130.85.97.55	228253
130.85.110.72	225304
130.85.84.224	224348
130.85.97.28	212550

Top Source IP - External

Source	No. of scan
213.180.193.68	51559
203.251.69.205	28392
61.146.52.26	28219
210.221.193.137	28190
138.100.42.180	27798
24.97.20.62	27447
194.79.163.149	27233
136.142.36.112	26338
205.118.75.10	26166
64.218.200.19	25657
148.235.166.150	25652
211.239.150.130	23864
81.255.41.226	23700
137.229.167.24	21976
210.96.67.220	19624

Top Destination IP - Internal

Source	No. of Scan
130.85.60.38	51601
130.85.97.87	32978
130.85.97.106	27348
130.85.97.88	12524
130.85.97.16	7245
130.85.97.21	4856
130.85.97.145	4613
130.85.97.104	4034
130.85.12.6	2684
130.85.12.6	2684

Top Destination IP - External

Source	No. of scan
69.6.57.4	107466
69.6.57.7	89702
69.6.57.9	89529
192.26.92.30	68417
192.48.79.30	55443
192.5.6.30	46521
69.6.57.8	45694
69.6.57.10	45498
195.228.156.17	44678
128.194.254.5	40203

It is note that the IP addresses in the scan files may not be obfuscated. '130.85' was not replaced with 'MY.NET' so it is pretty clear where all the log files are from.

Ports	No. of scan
53	3661660
135	3298396
25	754850
2745	564830
80	553356
6129	549850
3127	466572
445	461834
1025	449359
139	414950
3410	384789
5000	371532
22321	342977
4662	234439
6346	140305

3.6 OOS Log Analysis

From the analysis, there are 5,449 packets which can broken down into different out-of-packet flags below:

OOS	No. of OOS
12****S*	5152
12*A**S*	130
********	87
*2U*PRSF	2
**U*PRSF	2
12UAP*S*	1
12UAP***	1
12U*PRS*	1
12U*PR*F	1
12U*PR**	1

OOS	No. of OOS
12U*P*S*	1
12U***S*	1
12*APR*F	1
12*AP*SF	1
12*AP**F	1
12*A*R**	1
12**P*S*	1
12**P**F	1
1**A*RSF	1

OOS	No. of OOS
1****RSF	1
*2U***SF	1
*2*A*RSF	1
U*SF	1
U***	1
***A*RSF	1
Total	5449

The OOS log start with "12" or "*2" or "1*" mean RESERVED bits are set.

The first 3 oos flags account for 98% of the total packets. So I will be focusing on these 3 flags. [28]

Flag: 12****S* **No. of occurrences:** 5,152

This flag is generated when someone perform TCP SYN Scan to the network. SYN packet is used to initiate the three-way handshake. If the destination responses with SYN-ACK then it shows that the port is listening or if the destination responses with RST-ACK, then the port is not listening.

The attacker uses SYN Scan to find out about the listening ports of the target machine. Sample SYN Scans are :

```
04/12-00:06:43.044408 68.54.84.49:53332 -> MY.NET.6.7:110
TCP TTL:51 TOS:0x0 ID:27651 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x10163C88  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 914772107 0 NOP WS: 0

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

04/12-00:07:47.135360 68.54.84.49:53333 -> MY.NET.6.7:110
TCP TTL:51 TOS:0x0 ID:30136 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x15001C2C  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 914778516 0 NOP WS: 0
```

Flag: 12*AS*** **No. of occurrences:** 130

This is the SYN-ACK packet. Like I explained above, SYN-ACK is the response to the SYN initiation. It means that the server is alive and the requested port is listening.

The attacker can send the SYN-ACK without sending the SYN request. Usually the destination will response with RST-ACK to close the connection because this is not a complete three-way handshake. But this gives the information to the attacker that the host does alive and whether the post is closed.

Sample packets:

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

04/15-05:04:06.170426 61.184.240.120:80 -> MY.NET.120.56:61711
TCP TTL:112 TOS:0x0 ID:52573 IpLen:20 DgmLen:40 DF
12*A**S* Seq: 0x277B277B  Ack: 0x532A6D1E  Win: 0xFFFF  TcpLen: 20

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

04/15-05:04:13.236135 61.184.240.120:80 -> MY.NET.28.11:32106
TCP TTL:112 TOS:0x0 ID:4468 IpLen:20 DgmLen:40 DF
12*A**S* Seq: 0xE79BE79B  Ack: 0x7A353678  Win: 0xFFFF  TcpLen: 20

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

Flag: ******** **No. of occurrences:** 87

This is a NULL scan which TCP packet is sent with all flag turned off. The destination host usually send back a RST to all closed ports.

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

04/11-00:41:14.795330 68.121.194.43:6919 -> MY.NET.12.4:110
TCP TTL:78 TOS:0x0 ID:4660 IpLen:20 DgmLen:40
******** Seq: 0xFA88001  Ack: 0x547783B5  Win: 0x800  TcpLen: 20

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


04/11-01:03:09.443438 68.121.194.43:7175 -> MY.NET.12.4:110
TCP TTL:78 TOS:0x0 ID:4660 IpLen:20 DgmLen:40
******** Seq: 0xFC10001  Ack: 0xA817EEE8  Win: 0x800  TcpLen: 20

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

Other interesting OOS packets are XMAS and FULLXMAS scan:

04/07	16:41:28	82.64.84.58	0	130.85.42.3	0	XMAS	**U*P**F
04/07	22:08:33	61.216.77.99	8767	130.85.12.6	1063	XMAS	**U*P**F
04/08	02:45:38	4.8.204.245	33258	130.85.24.47	3964	XMAS	*2U*P**F
04/08	02:49:28	4.8.204.245	33258	130.85.24.47	3964	XMAS	*2U*P**F
04/08	03:31:26	4.8.204.245	33258	130.85.24.47	3964	XMAS	*2U*P**F
04/09	11:33:56	80.60.5.152	3869	130.85.111.34	40631	FULLXMAS	*2UAPRSF
04/10	02:40:01	212.76.225.24	19089	130.85.43.3	29333	FULLXMAS	*2UAPRSF
04/10	02:44:12	68.167.207.243	63885	130.85.153.35	3247	XMAS	*2U*P**F
04/10	11:45:06	68.108.222.13	6881	130.85.153.91	4112	XMAS	*2U*P**F
04/10	11:46:04	68.108.69.158	3726	130.85.5.20	80	XMAS	*2U*P**F
04/10	12:26:33	209.104.53.200	80	130.85.153.166	2285	XMAS	**U*P**F
04/10	12:27:03	66.43.22.192	1103	130.85.12.6	25	FULLXMAS	*2UAPRSF
04/10	12:29:55	209.104.53.200	33365	130.85.153.166	39334	XMAS	1*U*P**F
04/10	13:26:45	209.104.53.100	80	130.85.153.166	1778	XMAS	*2U*P**F

The XMAS scan turned on UPF flags and FULLXMAS scan turned on all flags. The purpose is to trick the system into responding to any of the requests. If the response is RST, it means port is not listening.

Most of the above source IPs appeared in alert files with different scan type alert messages. And some are result of the assembled Tiny Fragment packets (refer to 3.3.4 above)

Top source IP and destination IP from the OOS files are :

Top Source IP

Source IP	No. of scan
68.54.84.49	1621
202.54.60.162	253
66.225.198.20	154
62.174.236.17	154
61.184.240.120	130
212.202.14.132	110
68.55.57.217	108
193.170.194.27	81
68.121.194.43	79
216.95.201.21	72

Top Destination IP

Destination IP	No. of scan
MY.NET.12.6	1911
MY.NET.6.7	1637
MY.NET.60.14	219
MY.NET.24.44	206
MY.NET.12.4	135
MY.NET.43.3	134
MY.NET.5.67	117
MY.NET.70.164	110
MY.NET.29.3	98
MY.NET.43.2	87

3.7 Top Talkers

Below is a list of top talkers that generate alert messages, scan packets and out-of-alert packets. The top talkers are broken down into external hosts and internal hosts.

Alerts – Top External Talkers

Source IP	No. of alerts
212.76.225.24	7561
199.131.21.34	3480
68.81.0.87	2993
141.157.102.155	2693
131.92.177.18	2166
68.43.170.140	1566
69.138.77.62	1534
68.55.113.194	1483
68.57.90.146	1390
68.55.178.168	1298
138.88.183.54	1121
24.5.46.4	1105

Alerts – Top Internal Talkers

Source IP	No. of alerts
MY.NET.11.7	6734
MY.NET.84.235	3871
MY.NET.60.16	2169
MY.NET.111.228	991
MY.NET.150.198	634
MY.NET.150.44	619
MY.NET.97.51	618
MY.NET.75.13	578
MY.NET.11.6	524
MY.NET.97.92	379

Scan – Top External Talkers

Source	No. of scan
213.180.193.68	51559
203.251.69.205	28392
61.146.52.26	28219
210.221.193.137	28190
138.100.42.180	27798
24.97.20.62	27447
194.79.163.149	27233
136.142.36.112	26338
205.118.75.10	26166
64.218.200.19	25657

Scan – Top Internal Talkers

Source	No. of Scan
130.85.111.51	1621815
130.85.153.35	1522455
130.85.81.39	1187999
130.85.70.96	1130696
130.85.112.152	1082053
130.85.1.4	795875
130.85.66.56	334888
130.85.84.235	294843
130.85.42.2	253164
130.85.53.169	237464

148.235.166.150	25652
211.239.150.130	23864
81.255.41.226	23700
137.229.167.24	21976
210.96.67.220	19624

130.85.34.14	233328
130.85.97.55	228253
130.85.110.72	225304
130.85.84.224	224348
130.85.97.28	212550

OOS - Top Source IP

Source IP	No. of scan
68.54.84.49	1621
202.54.60.162	253
66.225.198.20	154
62.174.236.17	154
61.184.240.120	130
212.202.14.132	110
68.55.57.217	108
193.170.194.27	81
68.121.194.43	79
216.95.201.21	72

OOS - Top Destination IP

Destination IP	No. of scan
MY.NET.12.6	1911
MY.NET.6.7	1637
MY.NET.60.14	219
MY.NET.24.44	206
MY.NET.12.4	135
MY.NET.43.3	134
MY.NET.5.67	117
MY.NET.70.164	110
MY.NET.29.3	98
MY.NET.43.2	87

3.8 Top Targets

Alerts – Top External Targets

Destination IP	No. of alerts
169.254.0.0	5533
82.48.242.184	3240
141.157.102.155	2168
169.254.25.129	1755
24.5.46.4	1249
209.2.144.10	991
169.254.45.176	802
81.203.197.37	310
62.77.191.33	307
199.239.137.216	299

Alerts – Top Internal Targets

Destination IP	No. of alerts
MY.NET.30.3	12246
MY.NET.30.4	10074
MY.NET.43.3	7558
MY.NET.60.16	2693
MY.NET.84.235	1523
MY.NET.84.236	1056
MY.NET.17.3	850
MY.NET.70.74	832
MY.NET.153.83	615
MY.NET.1.3	599
MY.NET.97.51	510
MY.NET.84.204	448

3.9 Top Communications

Alerts – External -> Internal

Communications	No. of alerts
212.76.225.24->MY.NET.43.3	7544
68.81.0.87->MY.NET.30.4	2993
141.157.102.155->MY.NET.60.16	2693
131.92.177.18->MY.NET.30.3	2166
68.55.113.194->MY.NET.30.3	1483
69.138.77.62->MY.NET.30.3	1464
68.57.90.146->MY.NET.30.3	1355
68.55.178.168->MY.NET.30.3	1290
138.88.183.54->MY.NET.30.4	1121
68.55.62.244->MY.NET.30.4	858
199.131.21.34->MY.NET.84.235	791
199.131.21.34->MY.NET.84.236	767
151.196.115.104->MY.NET.30.3	751

Alerts – Internal -> External

Communications	No. of alerts
MY.NET.11.7>169.254.0.0	4996
MY.NET.84.235>82.48.242.184	3240
MY.NET.60.16>141.157.102.155	2168
MY.NET.11.7>169.254.25.129	1738
MY.NET.111.228>209.2.144.10	991
MY.NET.97.51>24.5.46.4	618
MY.NET.11.6>169.254.0.0	518
MY.NET.97.92>24.5.46.4	379
MY.NET.84.235>81.203.197.37	310
MY.NET.153.83>62.77.191.33	307
MY.NET.84.235>217.95.183.166	248
MY.NET.190.95>219.250.48.44	239
MY.NET.5.34>199.239.137.216	150
MY.NET.29.30>199.239.137.216	149

Top external scanners were selected base on number of occurrences. Information were extracted from different WHOIS web site such as http://ws.arin.net/cgi-bin/whois.pl, www.ripe.net,

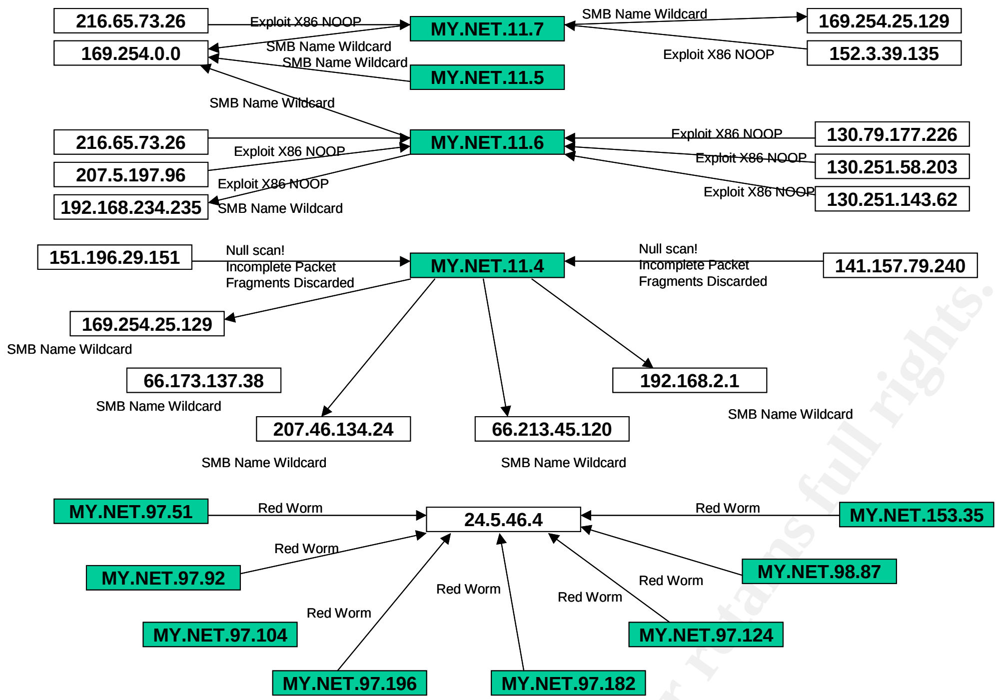
212.76.225.24 **7561 alerts**	**199.131.21.34** **3480 alerts**
inetnum: 212.76.225.0 - 212.76.225.255 netname: CODITEL descr: Coditel - Internet Services country: BE admin-c: XD6 tech-c: YB490-RIPE status: ASSIGNED PA notify: tech.registry@coditel.be mnt-by: CODITEL-MNT mnt-lower: CODITEL-MNT changed: xavier.darche@coditel.be 20010109 changed: xavier.darche@coditel.be 20030513 changed: xavier.darche@coditel.be 20030514 source: RIPE	OrgName: USDA Office of Operations OrgID: UOO-2 Address: Suite 133, Building A Address: 2150 Centre Ave City: Fort Collins StateProv: CO PostalCode: 80526 Country: US NetRange: 199.128.0.0 - 199.159.255.255 CIDR: 199.128.0.0/11 NetName: USDA-CBLK NetHandle: NET-199-128-0-0-1 Parent: NET-199-0-0-0-0 NetType: Direct Allocation NameServer: NS.USDA.GOV NameServer: NS2.USDA.GOV NameServer: NS3.USDA.GOV Comment: RegDate: 1994-02-08 Updated: 2000-06-16 TechHandle: ZU20-ARIN TechName: USDA - Office of the ChiefInformation Officer TechPhone: +1-970-295-5277 TechEmail: Network.Operations@usda.gov OrgAbuseHandle: ZU20-ARIN OrgAbuseName: USDA - Office of the ChiefInformation Officer OrgAbusePhone: +1-970-295-5277 OrgAbuseEmail: Network.Operations@usda.gov OrgNOCHandle: ZU20-ARIN OrgNOCName: USDA - Office of the ChiefInformation Officer OrgNOCPhone: +1-970-295-5277 OrgNOCEmail: Network.Operations@usda.gov OrgTechHandle: ZU20-ARIN OrgTechName: USDA - Office of the ChiefInformation Officer OrgTechPhone: +1-970-295-5277 OrgTechEmail: Network.Operations@usda.gov
68.81.0.87 2993 alerts	141.157.102.155 2693 alerts

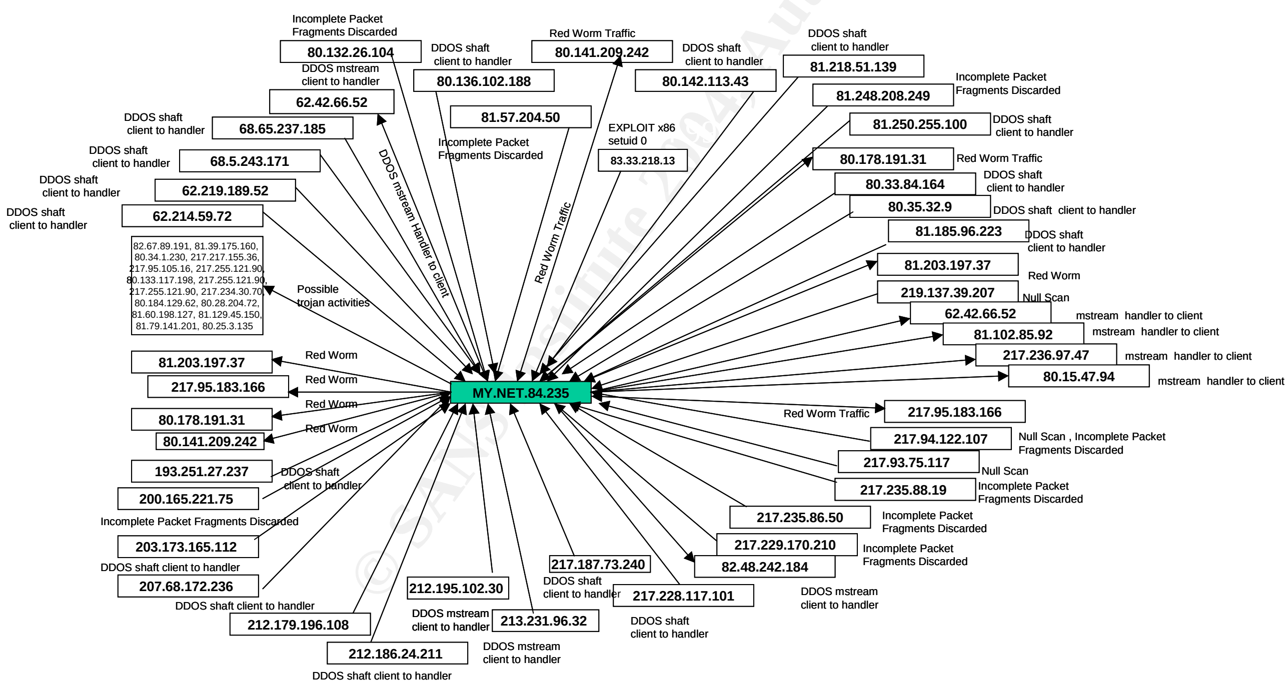
Comcast Cable Communications, Inc. JUMPSTART-2 (NET-68-80-0-0-1) 68.80.0.0 - 68.87.255.255 Comcast Cable Communications, Inc. PA-METRO-7 (NET-68-80-0-0-2) 68.80.0.0 - 68.81.255.255	Verizon Internet Services VIS-141-149 (NET-141-149-0-0-1) 141.149.0.0 - 141.158.255.255 Verizon Internet Services VZ-DSLDIAL-CYVLMD-9 (NET-141-157-57-0-1) 141.157.57.0 - 141.157.126.255
131.92.177.18 2166 alerts	
OrgName: Army Information Systems Command - Aberdeen (EA) OrgID: AISCAE Address: AMSSB-SCI-N/BLDG E5234 City: ABERDEEN PROVING GROUND StateProv: MD PostalCode: Country: US NetRange: 131.92.0.0 - 131.92.255.255 CIDR: 131.92.0.0/16 NetName: APGEA-NET1 NetHandle: NET-131-92-0-0-1 Parent: NET-131-0-0-0-0 NetType: Direct Assignment NameServer: NS01.ARMY.MIL NameServer: NS02.ARMY.MIL NameServer: NS03.ARMY.MIL Comment: RegDate: 1988-11-01 Updated: 2001-08-09 TechHandle: RW943-ARIN TechName: Ward, Ronnie TechPhone: +1-410-436-4755 TechEmail: RONNIE.WARD@sbccom.apgea.army.mil	

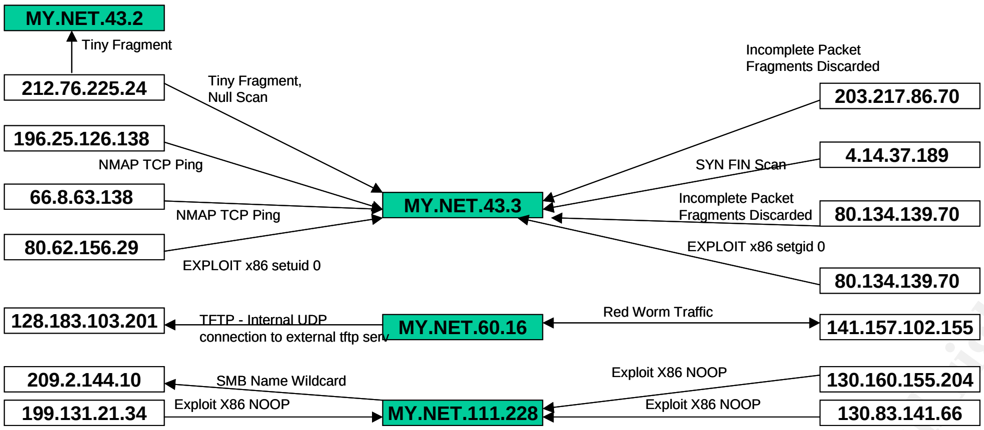
3.10 Link Graph

Link graph was created using top 5 source hosts and destination hosts that generate the alert messages. Source hosts and destination hosts include hosts inside and outside the network. The alert message is written near the box and the arrow is used to show direction of the traffic. All relevant information from OOS files and scan files were also used to develop the graph.

The box highlighted in green represents internal hosts and the box in white represents external host. The link graph shows the relationship between the top 5 talkers (both external and internal) and top 5 internal destinations.


216.65.73.26
Exploit X86 NOOP
MY.NET.11.7
SMB Name Wildcard
169.254.25.129
169.254.0.0
SMB Name Wildcard
SMB Name Wildcard
Exploit X86 NOOP
152.3.39.135
MY.NET.11.5
SMB Name Wildcard
216.65.73.26
Exploit X86 NOOP
MY.NET.11.6
Exploit X86 NOOP
130.79.177.226
207.5.197.96
Exploit X86 NOOP
130.251.58.203
192.168.234.235
Exploit X86 NOOP
SMB Name Wildcard
Exploit X86 NOOP
130.251.143.62
151.196.29.151
Null scan!
Incomplete Packet
Fragments Discarded
MY.NET.11.4
Null scan!
Incomplete Packet
Fragments Discarded
141.157.79.240
169.254.25.129
SMB Name Wildcard
66.173.137.38
SMB Name Wildcard
192.168.2.1
207.46.134.24
66.213.45.120
SMB Name Wildcard
SMB Name Wildcard
SMB Name Wildcard
MY.NET.97.51
Red Worm
24.5.46.4
Red Worm
MY.NET.153.35
Red Worm
MY.NET.97.92
MY.NET.98.87
Red Worm
Red Worm
Red Worm
Red Worm
MY.NET.97.104
MY.NET.97.124
MY.NET.97.196
MY.NET.97.182


Incomplete Packet
Fragments Discarded
80.132.26.104
DDOS shaft
client to handler
Red Worm Traffic
80.141.209.242
DDOS shaft
client to handler
DDOS shaft
client to handler
81.218.51.139
DDOS mstream
client to handler
80.136.102.188
80.142.113.43
62.42.66.52
81.248.208.249
Incomplete Packet
Fragments Discarded
DDOS shaft
client to handler
68.65.237.185
81.57.204.50
81.250.255.100
DDOS shaft
client to handler
DDOS shaft
client to handler
EXPLOIT x86
setuid 0
83.33.218.13
68.5.243.171
Incomplete Packet
Fragments Discarded
80.178.191.31
Red Worm Traffic
DDOS shaft
client to handler
62.219.189.52
80.33.84.164
DDOS shaft
client to handler
DDOS shaft
client to handler
62.214.59.72
80.35.32.9
DDOS shaft client to handler
DDOS mstream Handler to client
Red Worm Traffic
81.185.96.223
DDOS shaft
client to handler
82.67.89.191, 81.39.175.160,
80.34.1.230, 217.217.155.36,
217.95.105.16, 217.255.121.90,
80.133.117.198, 217.255.121.90,
217.255.121.90, 217.234.30.70,
80.184.129.62, 80.28.204.72,
81.60.198.127, 81.129.45.150,
81.79.141.201, 80.25.3.135
81.203.197.37
Red Worm
Possible
trojan activities
219.137.39.207
Null Scan
62.42.66.52
mstream handler to client
81.102.85.92
mstream handler to client
217.236.97.47
mstream handler to client
80.15.47.94
mstream handler to client
81.203.197.37
Red Worm
217.95.183.166
Red Worm
MY.NET.84.235
Red Worm
Red Worm Traffic
217.95.183.166
80.178.191.31
Red Worm
80.141.209.242
217.94.122.107
Null Scan , Incomplete Packet
Fragments Discarded
193.251.27.237
DDOS shaft
client to handler
217.93.75.117
Null Scan
200.165.221.75
217.235.88.19
Incomplete Packet
Fragments Discarded
Incomplete Packet Fragments Discarded
217.235.86.50
Incomplete Packet
Fragments Discarded
203.173.165.112
217.229.170.210
Incomplete Packet
Fragments Discarded
DDOS shaft client to handler
82.48.242.184
207.68.172.236
217.187.73.240
212.195.102.30
DDOS shaft
client to handler
217.228.117.101
DDOS mstream
client to handler
DDOS shaft client to handler
DDOS mstream
client to handler
213.231.96.32
DDOS shaft
client to handler
212.179.196.108
212.186.24.211
DDOS mstream
client to handler
DDOS shaft client to handler



Top internal destinations are MY.NET.30.3, MY.NET.30.4, MY.NET.43.3, MY.NET.60.16, and MY.NET.84.235. Although the number of alerts were mostly generated against MY.NET.30.3 and MY.NET.30.4 but the actual active target aimed by many external hosts is **MY.NET.84.235**. From the Link Graph and the analysis, looks like the host may have already been compromised. Immediate investigation is required.

3.11 List of Internal Hosts

Following is a list of hosts that might be infected with Trojan, worm or might have been compromised. I know that the list is long and there might be false as the analysis is done based on the limited information. However, it's worthwhile to have a look at these hosts.

3.11.1 Internal hosts that could be infected with sdbot

MY.NET.112.152	MY.NET.43.10
MY.NET.112.163	MY.NET.66.56
MY.NET.150.199	MY.NET.70.96
MY.NET.151.75	MY.NET.80.224
MY.NET.153.174	MY.NET.80.28
MY.NET.153.195	MY.NET.80.5
MY.NET.42.2	MY.NET.84.235
MY.NET.97.66	MY.NET.97.44
MY.NET.97.95	

3.11.2 Internal hosts that could be compromised with SUNRPC High Port Access

MY.NET.97.223	MY.NET.97.13	MY.NET.34.5	MY.NET.97.235

MY.NET.10.62	MY.NET.97.144	MY.NET.60.11	MY.NET.97.44
MY.NET.100.203	MY.NET.97.15	MY.NET.60.38	MY.NET.97.48
MY.NET.24.70	MY.NET.97.168	MY.NET.66.29	MY.NET.97.55
MY.NET.25.66	MY.NET.97.172	MY.NET.70.154	MY.NET.97.60
MY.NET.34.14	MY.NET.97.20	MY.NET.70.37	MY.NET.97.61
	MY.NET.97.213	MY.NET.82.106	MY.NET.97.94

3.11.3 Internal hosts that might be infected with Sub Seven or Ramen worm

MY.NET.16.106	MY.NET.190.100	MY.NET.190.110	MY.NET.190.12
MY.NET.16.114	MY.NET.190.101	MY.NET.190.111	MY.NET.190.120
MY.NET.16.90	MY.NET.190.102	MY.NET.190.112	MY.NET.190.121
MY.NET.190.0	MY.NET.190.103	MY.NET.190.113	MY.NET.190.122
MY.NET.190.1	MY.NET.190.104	MY.NET.190.114	MY.NET.190.123
MY.NET.190.10	MY.NET.190.105	MY.NET.190.115	MY.NET.190.124
MY.NET.190.13	MY.NET.190.106	MY.NET.190.116	MY.NET.190.125
MY.NET.190.130	MY.NET.190.107	MY.NET.190.117	MY.NET.190.126
MY.NET.190.131	MY.NET.190.108	MY.NET.190.118	MY.NET.190.127
MY.NET.190.132	MY.NET.190.109	MY.NET.190.119	MY.NET.190.128
MY.NET.190.133	MY.NET.190.11	MY.NET.190.15	MY.NET.190.129
MY.NET.190.134	MY.NET.190.140	MY.NET.190.150	MY.NET.190.8
MY.NET.190.135	MY.NET.190.141	MY.NET.190.151	MY.NET.190.80
MY.NET.190.136	MY.NET.190.142	MY.NET.190.152	MY.NET.190.81
MY.NET.190.137	MY.NET.190.143	MY.NET.190.153	MY.NET.190.82
MY.NET.190.138	MY.NET.190.144	MY.NET.190.154	MY.NET.190.83
MY.NET.190.139	MY.NET.190.145	MY.NET.190.155	MY.NET.190.84
MY.NET.190.14	MY.NET.190.146	MY.NET.190.156	MY.NET.190.85
MY.NET.190.16	MY.NET.190.147	MY.NET.190.157	MY.NET.190.86
MY.NET.190.160	MY.NET.190.148	MY.NET.190.158	MY.NET.190.87
MY.NET.190.161	MY.NET.190.149	MY.NET.190.159	MY.NET.190.88
MY.NET.190.162	MY.NET.190.19	MY.NET.190.18	MY.NET.190.89
MY.NET.190.163	MY.NET.190.190	MY.NET.190.180	MY.NET.190.9
MY.NET.190.164	MY.NET.190.191	MY.NET.190.181	MY.NET.190.90
MY.NET.190.165	MY.NET.190.192	MY.NET.190.182	MY.NET.190.91
MY.NET.190.166	MY.NET.190.193	MY.NET.190.183	MY.NET.190.92
MY.NET.190.167	MY.NET.190.194	MY.NET.190.184	MY.NET.190.93
MY.NET.190.168	MY.NET.190.195	MY.NET.190.185	MY.NET.190.94
MY.NET.190.169	MY.NET.190.196	MY.NET.190.186	MY.NET.190.95
MY.NET.190.17	MY.NET.190.197	MY.NET.190.187	MY.NET.190.96
MY.NET.190.170	MY.NET.190.198	MY.NET.190.188	MY.NET.190.97
MY.NET.190.171	MY.NET.190.199	MY.NET.190.189	MY.NET.190.98
MY.NET.190.172	MY.NET.190.21	MY.NET.190.23	MY.NET.190.99
MY.NET.190.173	MY.NET.190.210	MY.NET.190.230	MY.NET.5.5
MY.NET.190.174	MY.NET.190.211	MY.NET.190.231	MY.NET.6.15
MY.NET.190.175	MY.NET.190.212	MY.NET.190.232	MY.NET.190.60
MY.NET.190.176	MY.NET.190.213	MY.NET.190.233	MY.NET.190.61
MY.NET.190.177	MY.NET.190.214	MY.NET.190.234	MY.NET.190.62
MY.NET.190.178	MY.NET.190.215	MY.NET.190.235	MY.NET.190.63
MY.NET.190.179	MY.NET.190.216	MY.NET.190.236	MY.NET.190.64
MY.NET.190.2	MY.NET.190.217	MY.NET.190.237	MY.NET.190.65
MY.NET.190.20	MY.NET.190.218	MY.NET.190.238	MY.NET.190.66

MY.NET.190.200	MY.NET.190.219	MY.NET.190.239	MY.NET.190.67
MY.NET.190.201	MY.NET.190.22	MY.NET.190.24	MY.NET.190.68
MY.NET.190.202	MY.NET.190.220	MY.NET.190.240	MY.NET.190.69
MY.NET.190.203	MY.NET.190.221	MY.NET.190.241	MY.NET.190.7
MY.NET.190.204	MY.NET.190.222	MY.NET.190.242	MY.NET.190.70
MY.NET.190.205	MY.NET.190.223	MY.NET.190.243	MY.NET.190.71
MY.NET.190.206	MY.NET.190.224	MY.NET.190.244	MY.NET.190.72
MY.NET.190.207	MY.NET.190.225	MY.NET.190.245	MY.NET.190.73
MY.NET.190.208	MY.NET.190.226	MY.NET.190.246	MY.NET.190.74
MY.NET.190.209	MY.NET.190.227	MY.NET.190.247	MY.NET.190.75
MY.NET.190.25	MY.NET.190.228	MY.NET.190.248	MY.NET.190.76
MY.NET.190.250	MY.NET.190.229	MY.NET.190.249	MY.NET.190.77
MY.NET.190.251	MY.NET.190.3	MY.NET.190.4	MY.NET.190.78
MY.NET.190.252	MY.NET.190.30	MY.NET.190.40	MY.NET.190.79
MY.NET.190.253	MY.NET.190.31	MY.NET.190.41	MY.NET.190.5
MY.NET.190.254	MY.NET.190.32	MY.NET.190.42	MY.NET.190.50
MY.NET.190.26	MY.NET.190.33	MY.NET.190.43	MY.NET.190.51
MY.NET.190.27	MY.NET.190.34	MY.NET.190.44	MY.NET.190.52
MY.NET.190.28	MY.NET.190.35	MY.NET.190.45	MY.NET.190.53
MY.NET.190.29	MY.NET.190.36	MY.NET.190.46	MY.NET.190.54
MY.NET.190.58	MY.NET.190.37	MY.NET.190.47	MY.NET.190.55
MY.NET.190.59	MY.NET.190.38	MY.NET.190.48	MY.NET.190.56
MY.NET.190.6	MY.NET.190.39	MY.NET.190.49	MY.NET.190.57

3.11.4 Internal hosts that might be infected with mstream

MY.NET.60.17
MY.NET.84.235

3.12 Resources

[1] SANS. "Adore Worm". SANS Website. URL: http://www.sans.org/y2k/adore.htm (April 2001)
[2] Michael Murphy. "mstream – DDoS – Plain and Simple?". GIAC Website. URL: www.giac.org/practical/Michael_Murphy_GCIH.doc
[3] Brian Granier. "Intrusion Analysis and LaBrea Sentry". GIAC Website. URL: http://www.giac.org/practical/GCIA/Brian_Granier_GCIA.pdf. (October 2002)
[4] Giovanni Vigna. "phf Vulnerability". University of California Website. URL: http://www.cs.ucsb.edu/~vigna/courses/CS279/HW1/phf/.
[5] TonikGin. "XDCC – An .EDU Admin's Nightmare". NC State University Website. URL:http://www.ncsu.edu/it/security/papers/EduHacking.html (September 2002)
[6] Anonymous. "NETBIOS_NETBIOS-NT-NULL-SESSION". Digitrust Website. URL:http://www.digitaltrust.it/arachnids/IDS204/event.html
[7] Ricky Smith. "GCIA Practical Assignment". GIAC Website. URL: http://www.giac.org/practical/GCIA/Ricky_Smith_GCIA.pdf (December 2002)
[8] The Snort Project. Snort Users Manual 2.1.2. (March 2004)
[9] David Oborn. "GCIA Practical Assignment". GIAC Website. URL: http://www.giac.org/practical/David_Oborn_GCIA.html#detect4.
[10] David P. Reece. "Is blocking port 111 sufficient to protect your systems from RPC attacks?". SANS Website. URL: http://www.sans.org/resources/idfaq/blocking.php (February 2000)
[11] Terry MacDonald. "Intrusion Detection and Analysis: An Investigation ". GIAC Website. URL: http://www.giac.org/practical/GCIA/Terry_MacDonald_GCIA.pdf (June 2003)
[12] David Graham. "SANS2000 Certification Practical ". GIAC Website. URL: http://www.giac.org/practical/David_Graham.doc. (2000)
[13] Network Working Group. "Security Considerations for IP Fragment Filtering". Request for Comments: 1858. Andrew Systems Group Website. URL: http://asg.web.cmu.edu/rfc/rfc1858.html (October 1995)
[14] Anonymous. "TCP Port 27374". Link Logger Website. URL:http://www.linklogger.com/TCP27374.htm (February 2004)
[15] Anonymous. "Trojan list sorted on trojan port. " Simovits Consulting Website. URL: http://www.simovits.com/trojans/trojans.html
[16] Anonymous. "Linux.Ramen.Worm". Symantec Website. URL: http://securityresponse.symantec.com/avcenter/venc/data/linux.ramen.worm.html (2001)
[17] SANS. "Commonly Probed ports". SANS Website. URL: http://www.sans.org/y2k/ports.htm (May 2000)
[18] LURHQ Threat Intelligence Group. "Intrusion Detection – In-depth analysis". LURHQ Website. URL: http://www.lurhq.com/idsindepth.html

[19] Sami Rautiainen. "F-Secure Virus Descriptions : Adore" F-Secure Website. URL:http://www.f-secure.com/v-descs/adore.shtml (April 2001)
[20] Anonymous. "Voyager Alpha Force Distributed Denial of Service Agent". CanCERT Website. URL: http://www.cancert.ca/Alerts/AI-2001-11-22.pdf (November 2001)
[21] Anonymous. "Anti Trojan and Bot Scanner and Remover". SwatIT Website. URL: http://swatit.org/bots/
[22] Erik Montcalm. "GCIA Practical Assignment". GIAC Website. URL: http://www.secureops.com/papers/Erik_Montcalm_GSEC.pdf.
[23] Brian Caswell, Nigel Houghton. "Snort Signature Database". Snort Website. URL:http://www.snort.org/snort-db/sid.html?sid=1394
[24] Carlin Carpenter. "GCIA Practical Assignment". GIAC Website. URL:
[25] Anthony Dell. "Adore Worm – Another Mutation" GIAC Website. URL: http://www.giac.org/practical/gsec/Anthony_Dell_GSEC.pdf. (April 2001)
[26] Anonymous. "Linux.Ramen.Worm". Symantec Website. URL: http://securityresponse.symantec.com/avcenter/venc/data/linux.ramen.worm.html
[27] Anonymous. "FTP_DOS-FTPD-GLOBBING". Digitrust Website. URL: http://www.digitaltrust.it/arachnids/IDS487/event.html
[28] Ofir Arkin, "Network Scanning Techniques". Sys-Security Group Website.URL: http://www.sys-security.com/archive/papers/Network_Scanning_Techniques.pdf (November 1999)
[29] Richard Baker. "GCIA Practical Assignment". GIAC Website. URL: http://www.giac.org/practical/GCIA/Richard_Baker_GCIA.rtf. (November 2002)
[30] Dragos Ruiu. "spp_fnord: snort preprocessor - Multi-architecture mutated NOP sled detector". CANSECWEST Website. URL: http://www.cansecwest.com/spp_fnord.c (February 2002)
[31] Max Vision. " IDS177 "NETBIOS-NAME-QUERY". Digitrust Website. URL: http://www.whitehats.com/info/IDS177
[32] Network Working Group. "RFC 1918 - Address Allocation for Private Internets". Internet FAQ Archives Website. URL: http://www.faqs.org/rfcs/rfc1918.html (February 1996)
[33] Les M Gordon. "Intrusion Analysis - The Director's Cut!". GIAC Website. URL: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc (November 2002)
[34] Anonymous. "XDCC". 4Reference Website. URL:http://www.4reference.net/encyclopedias/wikipedia/XDCC.html
[35] Anonymous. "Windows VNC Server". RealVNC Website. URL:http://www.realvnc.com/winvnc.html