Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment
Version 3.4

Kam Hung Ng

May 2, 2004

Abstract

This version 3.4 assignment was written for the GCIA certification. It covers all the three parts as required.

Part 1 – Describe the state of Intrusion Detection

- Security Information Management (SIM) System

Part 2 – Network Detects

- Network Detect #1: Microsoft DCOM/RPC Buffer Overflow - Microsoft alert MS03-026
- Network Detect #2 : SHELLCODE x86 0xEB0C NOOP - ftp shellcode exploit
- Network Detect #3 : BAD-TRAFFIC ip reserved bit set

Part 3 – Analyze this

Date of log covered:

Scan logs:	**Alert logs:**	**Out-of-Spec logs:**
scans.040407	alert.040407	oos_report_040403
scans.040408	alert.040408	oos_report_040404
scans.040409	alert.040409	oos_report_040405
scans.040410	alert.040410	oos_report_040406
scans.040411	alert.040411	oos_report_040407

Security Information Management (SIM) System

Abstract

Today, corporations are commonly using point security solutions to monitor the security of their IS infrastructure such as firewalls, intrusion detection systems, proxies, screening routers, anti-virus system, anti-spam system, system monitoring software and other network monitoring devices.

There is one thing in common that they all can produce massive amount of alerts and logs. This was not a problem some years ago when the security landscape was still calm and quiet. But now, most networks are flooded with exploit traffic, improperly designed application traffic and other noises. This causes too many false positives to the security monitoring system which plagues the detection capability of real intrusions.

The common ways to handle these log data are 1) doing nothing; 2) manual data analysis, and 3) using in-house automation tools. [9] Often time, event information from only one source is not sufficient to get to the root cause of an alert. The security analyst will also have to refer to logs in other areas such firewall, system log and etc. This manual process could take a long time. Automation can help to improve the situation.

Organization is consistently under pressure to improve the bottom line and reduce cost. This will affect the resources available to security alert monitoring. How can the security team do more with less? The natural choice is to automate some of the time consuming and repetitive security tasks.

In response to these problems, security vendors develop better security management solutions. They are commonly referred to as Security Information Management (SIM) System.

Security Information Management (SIM) System

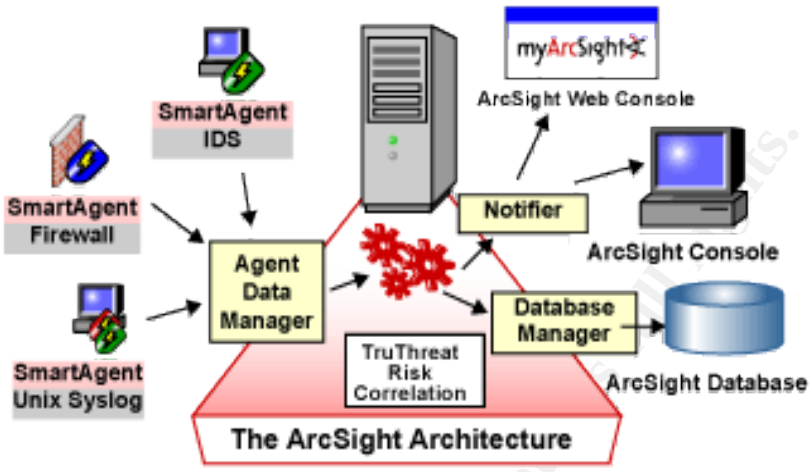
Richard Stiennon, Director, Gartner Group:

“Security event management solves todays' critical problem of aggregation and correlating diverse log data for real-time event detection and response. In the future I see these security management platforms serving as the central intelligence systems for security operations”

In essence, SIM acts as a hub of security information flow and provides a consistent analysis framework to produce near real time alert and long term threat trending report. The real time alert capability is critical for monitoring general exploit activities and the trending part has a better chance to detect

stealthy exploits.

SIM systems on the market today share similar architecture. Diagram below shows the basic ArcSight SIM system architecture. [v9]



System generally consists of the following basic components:

1. Event collector to collect the events from the security end-points such as IDS, firewalls, syslog and etc.
2. An event correlation engine to correlate the incoming events.
3. Database to store the event information.
4. Management and Alert console.
5. Data mining and reporting tools.

From functional perspective, SIM system has the following capabilities in general: [1]

1. Data Normalization – As SIM will take messages from security end-point systems of different suppliers with incompatible data format messages. The event collector of the SIM must be able to normalize the events to facilitate data process within the system.
2. Aggregation/Data Reduction – Consolidate data through deletion of duplicates, combining similar events into a single event.
3. Correlation Engine – The process of correlating related events into a single incident alert.
4. Data Transport – Moving log data around.
5. Alert Prioritization: Based on event severity, event decay time and other pre-defined parameters.
6. Response: This includes:
    - Automated notification to the console, pagers, and cell phones.
    - Incident Management system that keeps incident information and

provides flow management to the administrator to manage events.
- Knowledge Base of relevant industry, vendor, and organization policies to help event handling.
- Automatic and manual scripts/programs launching capabilities in response to an incident.

7. Data Mining: Capabilities to run report against the security event database to produce reports and metrics.

Event Correlation

Correlation is the key to SIM system. Correlation is a causal, complementary, parallel, or reciprocal relationship, especially a structural, functional, or qualitative correspondence between two comparable entities: a correlation between drug abuse and crime. [1]

The common types of correlation used are rule-based and statistical. [1][8] Rule-based correlation deals with states, conditions, time and other related actions/activities. Statistical(algorithmic) correlation employs special numeric algorithms to calculate threat levels of current attacks.

No one correlation method is better than the other. Statistical correlation system works like an anomaly detection system that it bases its decision on baseline information. While rule-based systems acts like a signature based intrusion detection system.

The datasets that SIM system correlates includes 1) security events and 2) system vulnerability database. Type 1 correlates only security events that SIM receives. Type 2 correlates incoming events with the information in the vulnerability database of the target system. SIM systems on the market use one type or both.

There are different level of correlations:

Atomic Level Correlation – The normalized raw event data from the security monitoring end-points are correlated. Often time the source IP, destination IP and the event class are used as the key correlation factors. Event class refers to a higher level of abstract event type. For example, for the same web server attack, IDS reports a “http_unicode_attack” event while the proxy server reports the same with different name “web_unicode_traffic”. In order to let the correlation engine to relate these two events, they have to be normalized with the same abstract event class name, say, “web_URL_attack” before sending it to the correlation engine. Of course, different vendor has different event class name. There are movement to unify this.[11]

Rule Correlation – An event may be tagged with time decay, severity decay and other attributes. It will be correlated based on the rule set of the correlation

engine. The value of these attributes will be adjusted by the correlation engine over time based on the algorithm used.

Vulnerability correlation – The process correlates the incoming event with a vulnerability database to determine the severity level of the event. The resulting event severity will be low if there is no vulnerabilities found on the target system for the current attack, for example, linux attack against a Window system.

Profile (Finger Print) Correlation [1] - Forensic network data such as remote port scans, remote OS finger prints, finger information, and banner snatching provides a series of data sets that can be compared to help correlate attacks to attacker profiles.

Watch List Correlation [1] – Using a set of learned inputs the watch list can serve as a reminder of previous offenders in real-time. Correlating previous attackers with current attacks.

SIM Can Reduce False Positive

SIM can reduce false positive mainly through atomic level, rule and vulnerability correlation.

Without correlation, all the event alerts from the point solution system will be sent to the corresponding monitor console, bombarding the administrator. The administrator can choose to trim down the amount of alerts by arbitrarily filtering off high noise alerts but taking the risks of missing critical and lethal events. It's a very difficult decision to make.

With atomic correlation, the SIM correlation engine inspects the incoming normalized raw event data from different end-points for:

1. Duplicated events. Duplicated events can come from the same sensor end-point or from different end-points. A lengthly network scan against a single system can generate a lot of IDS events. Normally, IDS systems already reduces the amount of alert event with threshold settings. But SIM system provides a second level of control to deal with the situation.

2. Related events. When a class C subnet was scanned by an attacker, it can generate roughly about 256K events assuming only the well-known low ports were scanned. Let say the IDS is stateful and only reports attacks against each system, you are still seeing 256 events on the IDS console. On the other hand, when these 256 host attack events were send to the SIM system, it can reduce to one single incident of attack class of a class C subnet scanning.

3. DDOS events. For normal IDS and firewall, they will produce large volume of

alerts as triggered by the DDOS event. But on a SIM system, it can reduce it down to a single incident warning that the victim was being attacked by many other systems.

With rule correlation, the SIM system correlation engine processes the incoming events based on time decay, severity decay and other attributes.

1. Time decay. The allows the correlation engine to accumulate events in the system until it reach the preset threshold before it sends alert to the console. This is helpful in reducing network activities like random probing. The attacker might just randomly send one or two probing packets to a systems and went away. Sending alerts for this kind of activities does not provide good value from intrusion monitoring perspective.

2. Severity decay. The severity of an alert will decay over time if the same alert does not happen again for a certain period of time. If this attack was really that lethal, the system should have been compromised at the first attack already. Therefore, if it does not come back within a preset period, the severity level should be reduced. Once it was reduced say from high to medium, the analyst will have one less alert to worry.

In my personal opinion, vulnerability correlation is one of the best ways to reduce false positive. It's a very straight forward approach. SIM system with this capability runs system vulnerabilities scanning at a preset schedule to perform security assessment on the systems it monitors. The results are stored to a central database. When the SIM system receives events, it checks them against the vulnerabilities database to see if the target system is vulnerable. Normally, if the target system is vulnerable to the attack, SIM will send an alert to the monitoring console. If no vulnerability is found for this exploit, it may just drop the incoming alert or only sends it to the alert repository for other use.

In essence, SIM system correlates information from multiple point detection systems running in solo to gain a better holistic view of the security context in the environment it protects, similar to the way an IDS analyst will do.

What Can We Do With SIM

Threat Analysis

One of the application of SIM is threat analysis. It is being used for near real time intrusion monitoring. Once an incident was alerted, either the system will fire off an automated response or the on-duty analyst will respond using the system.

Most of the IDS can do this. What makes SIM a better system?

1. SIM can reduce false positive.
2. It provides a more holistic view of the incident. Information pertaining to the event reported will be available to analyst at one single location presented in different view instead of the analyst has to look for information in several different systems. This speeds up the analysis process making the incident response time shorter.
3. Normally, SIM system presents a high level view of the incident alert that an analyst with less low level system knowledge is also able to handle the incident. This means that tight supply situation of qualified intrusion analyst will be somewhat improved. I am not here to suggest that the analysts are no longer needed to know the basic but with SIM system people somewhat less knowledge is still able to handle the common incidents.
4. SIM system provides flow management tool to help to manage the flow of incident handling and provides audit trail on the incident response. This will help to improve the performance of the security monitoring team.

Threat Trending

Most of the time, we are dealing with three major classes of exploits: noisy exploits, stealthy exploits and resource misuse. Noisy exploit activities can be handled by threat analysis discussed above. However, stealthy exploits and internal misuse often time requires to review tons of logs to discover. With the huge volume of logs, doing it manually or using once-awhile home grown scripts may not able to handle such a demand effectively and efficiently.

SIM system comes with canned data mining reporting modules for the user to use. It's relatively easy to show the top talkers, the top attackers, the top exploit class, the volume graph and etc.

But over time, custom reports will normally be needed to meet the unique needs of individual company. The information query is normally handled by database query scripts and reporting software such as SQL, Crystal report, PowerPlay and etc. Knowing how to use these tools is definite helpful to the analyst.

Trending provides valuable information about hidden security and non-security problems such as stealthy scan and exploits and someone has been spending most of the day in his/her office surfing the Internet.

Forensic Analysis

SIM system can help performing forensic analysis for system compromise or investigation. With the information stored in SIM system and the reporting and searching capabilities it provides, the analyst will be better equipped to handle computer security incidents. Forensic investigation normally requires reviewing load of historical data in a systematic manner. SIM system provides log data

storage and data mining capabilities for such purpose.

Policy Analysis

SIM system provides a more holistic view of the security posture of an organization. It gives a general overview of how the IS infrastructure is being used. With this information on hand, the organization can audit how well the corporate policies are being enforced and if the policies have to be adjusted to meet the consistent changing needs of the company.

Benefits of SIM

SIM system provides a centralized information management system as compared to the islands setup by individual point products. With SIM system, all events are displayed on centralized location/s with a unified view and the event data is kept in a central repository using the same data schema. Better quality information can be pulled from the SIM system to improve the decision-making process. This will also dramatically reduce the incident handling response time. All this will make the management of alerts, security incidents, reporting and metric generation more effective and efficient.

If the proper product is chosen and configured properly, false positive can be significantly reduced. This will put less strain on the analyst that they can focus on the real security issues.

After the SIM system has been configured and trained to do some of the repetitive incident handling work for the analysts, the efficiency of security operation will be increased, either operating resources can be reduced or the security operation team can do more in other areas to improve the organization security.

SIM system can provide better scalability. This is a big benefit to organization when more and more monitoring end-points are deployed to improve the security of the organization. Point solutions will not scale well in this kind of expansion requirements.

SIM systems currently on the market are able to work with third party security systems in existence today. Organization can leverage on the SIM system to make better utilization of existing security infrastructure and probably be able to extend their useful life to improve their return on investment. Also this allows customers to choose to use the the “Best of Breed” security products to make the best use of their security investment.

SIM System Selection Criteria

Architecture design

1. The architecture design of the product must provide good scalability that will meet the future plan of the organization.
2. The deployment and system maintenance requirements have to match with the organization requirements. Will this affect the service level of the organization during product installation and maintenance.
3. Does the product design follow good security guide lines and principles. I have seen security products with poor security design.
4. System redundancy.

Correlation

1. What correlation algorithm the product is using? Completely proprietary or using standard approach.
2. How difficult to make adjustment to the algorithm and the rules. I have seen product that requires actual configuration files and rule base file changes and yet the procedure is poorly documented. The system has to be reloaded even with minor changes. This may also cause problem in system software update/upgrade in the future. GUI approach sometimes make the adjustment easier and often it's less prone to incompatibility issues when changing version of the software as the users is isolated from making change to the core of the system directly.

System Configuration

1. How to configure the system.
2. Does the system provide configuration change audit and rollback capabilities.
3. Is system reboot required for configuration changes.

Third Party Product Support

1. List of support products.
2. Support information readily available and properly maintained.

Human Interface

1. How customizable is the administrator and monitoring console.
2. Does it have a reasonable layout?
3. Thin or thick client?
4. Security control of administrator and regular analysts access.
5. How many view of the same data is available.

Data Mining and Reporting

1. What database is supported?
2. Database access method?
3. Security control of the database.

Response Type

1. Send alert via email, pager, SNMP and other protocol.
2. Manual response, manual initiated script response, automated script response.
3. Sending response to other products.
4. Controllable from other systems.

Performance

1. System throughput.
2. Alert response time.

System Requirements

1. Operation System requirements. Does it use the standard OS platform in your organization.
2. Hardware requirements. Does it use the standard hardware platform in your organization.

System Maintenance

1. Regular system recycle requirement.
2. SIM software system patch notification mechanism.
3. Database maintenance utility included.

System Recovery

1. What to back up?
2. How often to backup?
3. System Recovery time?

Total Cost of Owner

1. Operation System software
2. Database license cost
3. Hardware cost
4. One time SIM software license cost
5. Recurring SIM software license cost

6. Maintenance charge
7. Installation cost
8. Training cost
9. administration cost

Conclusion

SIM systems will improve the effectiveness and efficiency of existing security monitoring and control operation if deployed properly. But it is still at its infancy stage. It will take many years to develop. How good it will end up is still a big unknown. The various correlation mechanism that comes with the product today is still very rudimentary that it is only good at detecting DDOS, network scan and similar real time attacks.

Setting up the system to monitor application level attack may not be a trivial task. SIM system relies on quality event inputs and the rule bases to make event correlation. Like a signature base IDS, it will not be able to detect security problem yet unknown to the administrator. It is definitely way more difficult and complicated than the story the vendor try to tell to convince you to open your wallet.

It is a real challenge for the IDS analysts to figure out ways to streamline and make the correlation more effective in detecting real intrusions over time in the dynamic security environment. In my personal opinion, I see that improvement and optimization in both the correlation engine and the input data are the top two factors critical to the successful implementation of SIM systems.

Like any other security products, it will never be a once-for-all situation. It has a life cycle needs to be maintained properly. Constantly baby sitting the system should be expected as the security landscape is constantly changing.

If the management thinks that SIM system can replace a seasoned IDS analyst because a buddy told him/her in the airport cafe, it's a very dangerous scenario.

Security is a human problem. Totally relying on machines to deal with human problems is bound for failure. History has proved this to us over and over again. I don't expect IT folks along can improve SIM systems to great extent. It needs support, knowledge and technology in various scientific disciplines in physics, signal processing, electronics engineering, computer science, sociology, cognitive, psychology, and evolutionary computing and worldwide cooperation.

Vendor Lists:
[v1] Network Intelligence: Network Intelligence Engine
URL: http://www.network-intelligence.com/products/
[v2] Netforsenics: Security Information Management (SIM)

URL: http://www.netforensics.com/documents/pr_sim.asp
[v3] Open : Threat manager
URL: http://www.open.com/
§§§§§[v4] Guardednet: neuSECURE
URL: http://www.guarded.net/prod.html
[v5] HP:
URL: http://h71028.www7.hp.com/enterprise/cache/7940-0-0-0-121.aspx
[v6] HigherTower: TowerView
URL: http://www.hightowersecurity.com/Unique.html
[v7] NetIQ: Incident and Event Management suite
URL: http://www.netiq.com/solutions/security/incident.asp
[v8] ArcSight: ArcSight product suite
URL: http://www.arcsight.com/product.htm
[v9] Symantec: Incident Manager
URL:
http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=166&EID=0
[v10] ISS: Site Protector
URL:
http://www.iss.net/products_services/enterprise_protection/rssite_protector/siteprotector.php
[v11] IBM: Risk Manager
URL: http://www-306.ibm.com/software/tivoli/products/risk-mgr/
[v12] Tenable: Lightning Console URL:
http://www.tenablesecurity.com/ids_fp.html
[v13] Intellitactics: Network Security Manager
URL: http://www.intellitactics.com/products/index.html
[v14] QuIDScor: Open source Qualys IDS Corrector Daemon
URL: http://quidscor.sourceforge.net/

Reference:

[1] GuardedNet Presentation:
URL: http://opensores.thebunker.net/pub/mirrors/blackhat/presentations/bh-usa-02/bh-us-02-caldwell-event.ppt
[2] NetIQ white paper:
URL:
http://download-src.netiq.com/Library/white_papers/Security_Event_Correlation-Where_Are_We_Now.pdf
[3] McIntyre, Kevin. “Event Correlation Systems”. 18 Feb 2003.
URL: http://www.giac.org/practical/GSEC/Kevin_McIntyre_GSEC.pdf
[4] Desai, Neil. “IDS Correlation of VA Data and IDS Alerts”. 30 Jun 2003.
URL: http://www.securityfocus.com/infocus/1708
[5] Caldwell, Matthew. “IDS correlation thread”. 22 Mar 2002.
URL: http://archives.neohapsis.com/archives/sf/ids/2002-q1/0403.html
[6] Dillis, Christoper. “IDS Event Correlation with SEC” 27 Jun 2003.

URL: http://www.giac.org/practical/GCIA/Christopher_Dillis_GCIA.pdf
[7] Simple Event Correlator:
URL: http://kodu.neti.ee/~risto/sec/
[8] Chuvakin, Anton. “Event Correlation in Security”. May 2003.
URL: http://www.tisc2001.com/newsletters/57.html
[9] Chuvakin, Anton. “Cross Platform Security Analysis”. May 2003.
URL: http://www.tisc2003.com/newsletters/418.html
[10] Moyer, Philip. “IDS correlation Thread” 22 Mar 2002.
URL: http://archives.neohapsis.com/archives/sf/ids/2002-q1/0410.html
[11] http://archives.neohapsis.com/archives/sf/ids/2002-q1/thread.html#401

Part 2 – Network Detect

Network Detect #1: Microsoft DCOM/RPC Buffer Overflow - Microsoft alert MS03-026

[**] [1:2351:1] NETBIOS DCERPC ISystemActivator path overflow attempt little endian [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
08/22/03-06:34:23.861089 0:D0:59:2B:7A:57 -> 0:50:DA:C5:9D:8B type:0x800 len:0x5EA
192.168.2.101:32777 -> 192.168.2.102:135 TCP TTL:64 TOS:0x0 ID:15427 IpLen:20 DgmLen:1500 DF
A* Seq: 0xBBCCB264 Ack: 0x3704D4B Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 167847 11838
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352]

Source of Trace
The raw log was download from http://isc.sans.org/logs/Raw/log_22_aug.raw.gz. All the analysis was done on a laptop running Fedora Linux (2.6.3 kernel). In order to aid the analysis of the detect, I wrote several scripts. To use these scripts, I exported the raw log file to text format file first by running the following commands.

- gunzip log_22_aug.raw.gz
- tcpdump -r log_22_aug.raw -e >> log.txt

Scanlog [20] script was used to summarize the traffic flow and to guess network devices in the system. Network devices like router, firewall and gateway will normally have more than one IP address associated with their MAC address. The oui.txt from IEEE (http://standards.ieee.org/regauth/oui/oui.txt) was also used as another indicator for network equipment. If the NIC belongs to Cisco, it has higher chance that it's a network device.

Network traffic summary (<N-D> after the MAC address indicates that the NIC is very likely belong to a network device):

00:04:e2:49:01:5e(SMCNetworks,Inc.)<N-D>|207.68.171.244|-->|
00:50:da:c5:9d:8b(3COMCORPORATION)|192.168.2.102
00:04:e2:49:01:5e(SMCNetworks,Inc.)<N-D>|208.172.13.190|-->|
00:50:da:c5:9d:8b(3COMCORPORATION)|192.168.2.102
00:50:da:c5:9d:8b(3COMCORPORATION)|192.168.2.102|-->|00:04:e2:49:01:5e
(SMCNetworks,Inc.)<N-D>|207.68.171.244
00:50:da:c5:9d:8b(3COMCORPORATION)|192.168.2.102|-->|00:04:e2:49:01:5e
(SMCNetworks,Inc.)<N-D>|208.172.13.190
00:50:da:c5:9d:8b(3COMCORPORATION)|192.168.2.102|-->|00:d0:59:2b:7a:57
(AMBITMICROSYSTEMSCORP.)|192.168.2.101
00:d0:59:2b:7a:57(AMBITMICROSYSTEMSCORP.)|192.168.2.101|-->|
00:50:da:c5:9d:8b(3COMCORPORATION)|192.168.2.102

All Mac and IP inventory list (src and dst: NIC used as source and destination respectively):
00:04:e2:49:01:5e(SMCNetworks,Inc.)<N-D>|207.68.171.244|dst
00:04:e2:49:01:5e(SMCNetworks,Inc.)<N-D>|207.68.171.244|src
00:04:e2:49:01:5e(SMCNetworks,Inc.)<N-D>|208.172.13.190|dst
00:04:e2:49:01:5e(SMCNetworks,Inc.)<N-D>|208.172.13.190|src
00:50:da:c5:9d:8b(3COMCORPORATION)|192.168.2.102|dst
00:50:da:c5:9d:8b(3COMCORPORATION)|192.168.2.102|src
00:d0:59:2b:7a:57(AMBITMICROSYSTEMSCORP.)|192.168.2.101|dst
00:d0:59:2b:7a:57(AMBITMICROSYSTEMSCORP.)|192.168.2.101|src

MAC/IP duplicate use statistics (The number in the third column separated by | tells how many IP were used with the MAC address. More than one will normally indicates a network devices such as router, firewall or gate.):

multi ip|src mac|2|00:04:e2:49:01:5e(SMCNetworks,Inc.)
multi ip|dst mac|2|00:04:e2:49:01:5e(SMCNetworks,Inc.)

From the data collected above, I believe this is a home network with a SMC firewall/router connected to the Internet. The default IP address of the SMC router inside interface is 192.168.2.0/24. There are two desktops sitting on the inside. The IDS was sitting on the internal network.

```
                ( Internet )
                     |
[ SMC FW/Router ] 00:04:e2:49:01:5e          <- Internet Gateway
                     |
[PC 192.168.2.101/00:d0:59:2b:7a:57]         <- Internal network
[PC 192.168.2.102/00:50:da:c5:9d:8b]         <- Internal network
[IDS]                                        <- Internal network
```

Script scan_allow2 [22] checks the allow inbound and outbound traffic to gain a better understanding about the environment. I used the following crude criteria for checking:

1. Allow outbound = syn packet out and there is syn/ack packet in for the same source and destination IP address combination. Syn and ack seq numbers are not tracked to simplify the program. This is why I said it is a crude method. But I believe this is good enough for this purpose.
2. Allow inbound = syn packet allow in.

The result indicates that only outbound http 80 traffic was generated. No outside initiated inbound traffic was reported. This combination matches quite well with home use router setting.

Detect was generated by:
Snort Version 2.1.2 (Build 25) was used to generate the alert. The following command line was used:

snort -r log_22_aug.raw -h 192.168.2.0/24 -l log -k none -c snort.conf -yeX

Switch -k was used because the raw log checksum has been modified.

The snort.conf has all the rules selected and the flow-portscan preprocessor turned on.

Snort created directory 192.168.2.101 and alert file. Under 192.168.2.101 directory, there are two files TCP:32777-135 and TCP:32778-53.

Script scanalertip [23] was then run against the 'alert' file to summarize the attack signature and IP address information. The output is listed immediately below. The leftmost number is the signature hit count.

3 [**] ATTACK-RESPONSES directory listing [**]|192.168.2.102:53|->|
192.168.2.101:32778|TCP
1 [**] NETBIOS DCERPC ISystemActivator path overflow attempt little endian [**]|192.168.2.101:32777|->|192.168.2.102:135|TCP

I picked 'NETBIOS DCERPC ISystemActivator path overflow attempt little

endian' alert as it is an interesting one to me.

This alert was generated by the following snort rule.

alert tcp $EXTERNAL_NET any -> $HOME_NET 135 (msg:"NETBIOS DCERPC ISystemActivator path overflow attempt little endian"; flow:to_server,established; content:"|05|"; distance:0; within:1; byte_test:1,&,16,3,relative; content:"|5c 00 5c 00|"; byte_test:4,>,256,-8,little,relative; reference:cve,CAN-2003-0352; classtype:attempted-admin; sid:2351; rev:1;)

This is not a simple rule that deserves some explanation. Snort Users manual 2.1.1 is used as reference.

1. Flow rule option 'to_server': only traffic flow to server will trigger this rule.
2. First content check: a value of 0x05 (0x = hex) has to be at the first byte of the payload data.
3. Second content check: a value of 0x10 has to be at the fourth (offset of 3) byte after the first 0x05 detected.
4. Then, it looks for pattern match of 0x5c005c00.
5. Lastly, check 4 bytes at a time for value greater than 256, starting 8 bytes before the match of item 4 above. These bytes will be processed in little endian fashion (Windows system data memory storage standard.

Actual captured traffic with some payload data to illustrate the rule triggering points:

[**] NETBIOS DCERPC ISystemActivator path overflow attempt little endian [**]
08/22/03-06:34:23.861089 0:D0:59:2B:7A:57 -> 0:50:DA:C5:9D:8B type:0x800 len:0x5EA
192.168.2.101:32777 -> 192.168.2.102:135 TCP TTL:64 TOS:0x0 ID:15427 IpLen:20 DgmLen:1500 DF
A* Seq: 0xBBCCB264 Ack: 0x3704D4B Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 167847 11838

```
0x0000   4500 05dc 3c43 4000 4006 72bd c0a8 0265      E...<C at .
0x0010   c0a8 0266 8009 0087 bbcc b264 0370 4d4b      ...f.......d.pMK
0x0020   8010 16d0 8454 0000 0101 080a 0002 8fa7      .....T.........
0x0030   0000 2e3e [05(2)] 00 0003 [10(3)] 00 0000 a806 0000      ...>.............
```

NOTE:
1. [05(2)] marked above meets the matching criteria #2 as it's the first byte of value 0x05 of the payload right after 20 bytes of IP and 32 bytes of TPC headers.
2. [10(3)] marked above meets the matching criteria #3 as it's third bytes counting from 0 after the matching of [05(20].

<snip>
0x0390 0000 0000 8601 0000 0000 [0000 8601(5)] 0000 .....
0x03a0 [5c00 5c00(4)] 4600 5800 4e00 4200 4600 5800 \.\.F.X.N.B.F.X.
0x03b0 4600 5800 4e00 4200 4600 5800 4600 5800 F.X.N.B.F.X.F.X.

NOTE:
1. [5c00 5c00(4)] matches the fourth criteria.
2. [0000 8601(5)] matches the last criteria. Of course I assume an offset of -8 is valid.

Probability the source address was spoofed:

The source must be the true IP address as:
1. This attack used TCP.
2. The attacker was actually controlling the system using the back door shell after the compromise was successful. The attacker's action will be described briefly below in the 'Attack mechanism' section.
3. There are only two machines involved in this scenario. Traffic analysis does not indicate there's other system involved that could be used as an intermediary system.

Description of Attack:
The attack was originated from system with IP 192.168.2.101 sitting on the inside network of the home system same as the victim Windows system with IP 192.168.2.102. The attack targeted a critical Windows Distributed Component Object Model (DCOM) Remote Procedure Call (RPC) interface vulnerability (MS03-026: http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx) existed in the victim machine. The target port used to exploit the buffer overflow vulnerability was TCP 135. The back door port on the victim system was chosen by the attacker at TCP port 53.

I believed that the attack was based on a C program called dcom.c or its variants. Paper of both Brian [2.2] and Wayne [2.1] mention other variants.

The dcom.c exploit program is at http://www.metasploit.com/tools/dcom.c.

The header section of the program is listed below for discussion.
```
/*
DCOM RPC Overflow Discovered by LSD
-> http://www.lsd-pl.net/files/get?WINDOWS/win32_dcom
 Based on FlashSky/Benjurry's Code
-> http://www.xfocus.org/documents/200307/2.html
 Written by H D Moore <hdm [at] metasploit.com>
-> http://www.metasploit.com/
```

```
 - Usage: ./dcom <Target ID> <Target IP>
- Targets:
-        0   Windows 2000 SP0 (english)
-        1   Windows 2000 SP1 (english)
-        2   Windows 2000 SP2 (english)
-        3   Windows 2000 SP3 (english)
-        4   Windows 2000 SP4 (english)
-        5   Windows XP SP0 (english)
-        6   Windows XP SP1 (english)
*/
```

It can be seen that this exploit targets only Windows systems. It will adjust itself for different service pack level of the target system.

Once compiled, it is very simple and easy to use. For example, 'dcom 3 192.168.2.102' will compromise the vulnerable system 192.168.2.102 running Windows 2000 with SP3(Service Pack 3).

This exploit starts up a command shell at the target system as determined by the 'target_ip.sin_port = htons(4444);' statement in the program. The default in the program is 4444 but it can be changed to any port by the attacker. For this detect, it's using port 53.

Attack Mechanism:
Based on the analysis of Wayne [2.1] and Brian [2.2], the exploit targeted the 'szName' buffer in the DCOM function 'CoGetInstanceFromFile'. It connects to TCP port 135 of a vulnerable victim machine to issue an MSRPC request for the file '\\servername\c$\123456111111111111111.doc' to overflow the 'szName' buffer to run its own code in the payload to spawn a command shell on a port defined by the attacker. It's tcp port 53 in this case.

The attack was done in the following sequence starting at 06:34:19. Note that 192.168.2.101 is the attacker system.

- 192.168.2.101:32777 -> 192.168.2.102:135 S <- Start 3 way TCP, handshake on port 135 used by Microsoft RPC
- 192.168.2.102:135 -> 192.168.2.101:32777 S/A
- 192.168.2.101:32777 -> 192.168.2.102:135 ./A
- 192.168.2.101:32777 -> 192.168.2.102:135 P/A
- 192.168.2.102:135 -> 192.168.2.101:32777 ./A
- 192.168.2.102:135 -> 192.168.2.101:32777 P/A
- 192.168.2.101:32777 -> 192.168.2.102:135 ./A
- 192.168.2.101:32777 -> 192.168.2.102:135 ./A <- Run buffer Overflow in this data packet

```
0x0000   4500 05dc 3c43 4000 4006 72bd c0a8 0265        E...<C at .
```

```
0x0010   c0a8 0266 8009 0087 bbcc b264 0370 4d4b      ...f.......d.pMK
0x0020   8010 16d0 8454 0000 0101 080a 0002 8fa7      .....T..........
0x0030   0000 2e3e 0500 0003 1000 0000 a806 0000      ...>............
0x0040   e500 0000 9006 0000 0100 0400 0500 0600      ................
<snip>
0x0390   0000 0000 8601 0000 0000 0000 8601 0000      ................
0x03a0   5c00 5c00 4600 5800 4e00 4200 4600 5800      \.\.F.X.N.B.F.X.
0x03b0   4600 5800 4e00 4200 4600 5800 4600 5800      F.X.N.B.F.X.F.X.
0x03c0   4600 5800 4600 5800 9f75 1800 cce0 fd7f      F.X.F.X..u......
0x03d0   cce0 fd7f 9090 9090 9090 9090 9090 9090      ................
0x03e0   9090 9090 9090 9090 9090 9090 9090 9090      ................
0x03f0   9090 9090 9090 9090 9090 9090 9090 9090      ................
```

NOTE: classical NOOP 0x90 data pattern found in Intel system Buffer Overflow showed up here ...
<snip>
NOTE: Blow is the buffer overflow attack code..

```
0x0470   9090 9090 9090 9090 9090 90eb 195e 31c9      ............^1.
0x0480   81e9 89ff ffff 8136 80bf 3294 81ee fcff      .......6..2.....
0x0490   ffff e2f2 eb05 e8e2 ffff ff03 5306 1f74      ............S..t
0x04a0   5775 9580 bfbb 927f 895a 1ace b1de 7ce1      Wu.......Z....|.
<snip>
0x05a0   8fb1 78d4 320e b0b3 7f01 5d03 7e27 3f62      ..x.2.....].~'?b
0x05b0   42f4 d0a4 af76 6ac4 9b0f 1dd4 9b7a 1dd4      B....vj......z..
0x05c0   9b7e 1dd4 9b62 19c4 9b22 c0d0 ee63 c5ea      .~...b..."...c..
0x05d0   be63 c57f c902 c57f e922 1f4c                .c.......".L
```

- 192.168.2.101:32777 -> 192.168.2.102:135 P/A

```
0x0000   4500 0134 3c44 4000 4006 7764 c0a8 0265      E..4<D at .
0x0010   c0a8 0266 8009 0087 bbcc b80c 0370 4d4b      ...f.........pMK
0x0020   8018 16d0 3ad9 0000 0101 080a 0002 8fa7      ....:...........
<snip>
0x00c0   e4f0 9080 2fa2 0400 5c00 4300 2400 5c00      ..../...\.C.$.\.
0x00d0   3100 3200 3300 3400 3500 3600 3100 3100      1.2.3.4.5.6.1.1.
0x00e0   3100 3100 3100 3100 3100 3100 3100 3100      1.1.1.1.1.1.1.1.
0x00f0   3100 3100 3100 3100 3100 2e00 6400 6f00      1.1.1.1.1...d.o.
0x0100   6300 0000 0110 0800 cccc cccc 2000 0000      c...............
0x0110   3000 2d00 0000 0000 882a 0c00 0200 0000      0.-......*......
0x0120   0100 0000 288c 0c00 0100 0000 0700 0000      ....(...........
0x0130   0000 0000                                    ....
```

NOTE: The 123456111111111111111.doc was pushed over to the victim in the packet above.

- 192.168.2.101:32777 -> 192.168.2.102:135 F/A
- 192.168.2.102:135 -> 192.168.2.101:32777 ./A
- 192.168.2.102:135 -> 192.168.2.101:32777 F/A

192.168.2.101:32777 -> 192.168.2.102:135 . /A <- graceful termination

NOTE: Accessing the newly established back door on the victim machine..

- 192.168.2.101:32778 -> 192.168.2.102:53 S <- access the back door
- 192.168.2.102:53 -> 192.168.2.101:32778 S/A

<snip>

- 192.168.2.102:53 -> 192.168.2.101:32778 P/A

```
0x0000   4500 005e 008e 4000 8006 73f0 c0a8 0266    E..^.. at ...s....f
<http://www.dshield.org/mailman/listinfo/intrusions>
0x0010   c0a8 0265 0035 800a 0383 8fb3 bc33 298d    ...e.5.......3).
0x0020   8018 4470 2ad2 0000 0101 080a 0000 2e61    ..Dp*..........a
0x0030   0002 900c 4d69 6372 6f73 6f66 7420 5769    ....Microsoft.Wi
0x0040   6e64 6f77 7320 3230 3030 205b 5665 7273    ndows.2000.[Vers
0x0050   696f 6e20 352e 3030 2e32 3139 355d         ion.5.00.2195]
```

NOTE: The victim system displayed system prompt. At this point, the attacker was having full access to the system such as listing files on the system.

```
0x0170   2020 2020 3c44 4952 3e20 2020 2020 2020    ....<DIR>.......
0x0180   2020 2050 726f 6772 616d 2046 696c 6573    ...Program.Files
0x0190   0d0a 3038 2f30 312f 3230 3033 2020 3131    ..08/01/2003..11
0x01a0   3a32 3561 2020 2020 2020 2020 2020 2020    :25a............
0x01b0   2033 3937 2c33 3132 2057 696e 4475 6d70    .397,312.WinDump
0x01c0   2e65 7865 0d0a 3038 2f31 312f 3230 3033    .exe..08/11/2003
0x01d0   2020 3039 3a33 3061 2020 2020 2020 3c44    ..09:30a......<D
```

NOTE: And the attacker was now able to run Windump.

```
0x0030   0002 b28f 7769 6e64 756d 702e 6578 6520    ....windump.exe.
0x0040   2d6e 5876 7320 3020 6970 2061 6e64 2068    -nXvs.0.ip.and.h
0x0050   6f73 7420 3139 322e 3136 382e 322e 3130    ost.192.168.2.10
0x0060   3220 616e 6420 7463 700a                   2.and.tcp.
<snip>
0x0000   4500 0082 00b5 4000 8006 73a5 c0a8 0266    E..... at ...s....f
0x0010   c0a8 0265 0035 800a 0383 9896 bc33 29e7    ...e.5.......3).
0x0020   8018 4416 46e4 0000 0101 080a 0000 31bc    ..D.F.........1.
0x0030   0002 b28f 7769 6e64 756d 702e 6578 653a    ....windump.exe:
0x0040   206c 6973 7465 6e69 6e67 206f 6e20 5c44    .listening.on.\D
0x0050   6576 6963 655c 4e50 465f 7b37 4630 3244    evice\NPF_{7F02D
0x0060   3334 362d 4342 3039 2d34 3139 332d 4232    346-CB09-4193-B2
```

```
0x0070   3343 2d43 3542 3337 3244 3441 3741 417d      3C-
                                                      C5B372D4A7AA}
<snip>
0x0160   2e2e 0d0a 3078 3030 3230 0920 3730 3032      ....0x0020..7002
0x0170   2034 3030 3020 3765 3936 2030 3030 3020      .4000.7e96.0000.
0x0180   3032 3034 2030 3562 3420 3031 3031 2030      0204.05b4.0101.0
0x0190   3430 3209 702e 402e 7e2e 2e2e 2e2e 2e2e      402.p. at .
0x01a0   2e2e 2e2e 0d0a 3038 3a33 363a 3033 2e30      ......08:36:03.0
0x01b0   3835 3335 3920 4950 2028 746f 7320 3078      85359.IP.(tos.0x
0x01c0   302c 2074 746c 2034 382c 2069 6420 3334      0,.ttl.48,.id.34
0x01d0   3037 392c 206c 656e 2034 3829 2032 3037      079,.len.48).207
0x01e0   2e36 382e 3137 312e 3234 342e 3830 203e      .68.171.244.80.>
0x01f0   2031 3932 2e31 3638 2e32 2e31 3032 2e31      .192.168.2.102.1
0x0200   3034 313a 2053 205b 7463 7020 7375 6d20      041:.S.[tcp.sum.
0x0210   6f6b 5d20 3232 3932 3933 3538 3536 3a32      ok].2292935856:2
0x0220   3239 3239 3335 3835 3628 3029 2061 636b      292935856(0).ack
0x0230   2038 3335 3930 3337 3420 7769 6e20 3137      .83590374.win.17
0x0240   3532 3020 3c6d 7373 2031 3436 302c 6e6f      520.<mss.1460,no
0x0250   702c 6e6f 702c 7361 636b 4f4b 3e20 2844      p,nop,sackOK>.(D
```

NOTE: Windump was successfully run as shown above. At this point, there's no question that the attacker was the co-owner of the victim system.

Correlation
The alerts were posted at a number of websites. I just choose same to list:

1. http://xforce.iss.net/xforce/xfdb/12629
2. http://www.cert.org/advisories/CA-2003-16.html
3. http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx
4. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352
5. http://isc.incidents.org/diary.php?date=2003-08-01

The Microsoft DCOM/RPC buffer overflow exploit was discussed in details in the following Internet sites:
[2.1] Freeman, Wayne. “An Analysis of the Microsoft RPC/DCM vul”. 22 Sep 2003.
URL: http://www.inetsecurity.info/downloads/papers/MSRPCDCOM.pdf
[2.2] Porter, Brian. “RPC-DCOM Vulnerability & Exploit”. 2 Nov 2003.
URL: http://www.giac.org/practical/GCIH/Brian_Porter_GCIH.pdf
[2.3]. alt.don. “Cursory analysis of the RPC DCOM exploit rewritten by hdm”. 6 Aug 2003.
URL: http://www.helpforums.co.uk/forum/viewtopic.php?t=7341.
[2.4] Counterpane Security Alerts, Microsoft RPC DCOM Remote Shell Vulnerabilities, 1 Aug 2003.
URL: http://www.counterpane.com/alert-v20030801-001.html

The exploit in this analysis used TCP 53 for the back door on the victim system instead of port 4444 as generally mentioned in the websites above. This can be changed very easily in the dcom.c C program. All the attacker has to do is to change the port parameter in statement 'target_ip.sin_port = htons(4444);' and recompile the C code. I am not suggesting that this exploit must have the port changed this way but it's a possibility. As Brian [2.2] mentioned in his paper that some exploit variants allow attacker to specify port at run time.

Evidence of Active targeting
It is evident that the attacker at 192.168.2.101 was actively targeting the MSRPC vulnerability in victim machine 192.168.2.102 at TCP port 135. After the exploit was successful, the attacker poked around the system and ran windump. There's no other system on the same subnet for the attacker to target.

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

5 = (3 + 5) - (2 + 1)

Criticality = 3. It's a home system and there are only two, I personally would be very concerned about any potential system compromise. But still it is not the end of the world. So I give it a three.

Lethality = 5. The attacker owned the system already and was having full access.

System Countermeasures = 2. Obviously, the system was missing patch to allow the exploit. I rate it on the safe side as I don't have other information to properly assess the situation.

Network Countermeasures = 1. The intruder and the victim systems are running on the same subnet. Normally, there will be no firewall in between. So there's basically no network countermeasures.

A score of severity of 5 is a very bad security situation that immediate action must be taken to mitigate the risks.

Defensive Recommendation
We have to address the following:
1. How to prevent the victim system from future exploits?
2. How can a system on the internal trusted subnet become an evil machine?

For point #1:
1. I would like to shut down MSRPC but I don't think this is a practical option with

Windows system.
2. The most important measure is to keep the system patch level up to date. Every single web page I came cross that relates to this exploit urges user to do so. And from a system security vulnerability perspective, this is the most effective mitigation and most often is the only resolution.

For point #2:
1. System access control has to be reviewed and further tightened up if deemed necessary.
2. The policy guards against system misuse must be reviewed. A new one must be written immediately if not in place yet.
3. Communicate the policy to the users to raise their security awareness.

If the above recommendations can be carried out, I will give both system and network countermeasures at least a four. I did not recommend any network security change, why I give network countermeasure a four also. It's because in this situation to deal with an internal network, a good security policy and user awareness should be able to improve the situation and adding a firewall is not a practical solution.

With these countermeasure improvement in place, we can re-calculate the severity.

Severity = (3 + 5) - (4 + 4) = 0, a significant improvement.

Multiple choice test question
Why 0x90 is normally seen in a buffer overflow exploiting Intel based systems?
a. It's a random data filler in the data stream helps to overflow the target buffer.
b. It's the procedure header preamble in Windows system.
c. Loading some 0x90 (NOP in Intel system) data pattern before the buffer overflow code ease the prediction of the program return address in a buffer overflow attempt.
d. It must be loaded into the buffer otherwise the buffer overflow attempt will fail.

Answer: c.

As quoted from http://www.securiteam.com/securityreviews/5OP0B006UQ.html

'Well, this is the top of the stack at crash time. It is safe to presume that we can use this as return address to our shellcode. We will now add some NOP (no operation) instructions before our buffer, so we do not have to be 100% correct regarding the prediction of the exact start of our shellcode in memory (or even brute forcing it).'

Intrusions@incidents.org mailing list posting

I posted this detect twice and have not received any questions or comments so far.

First post: [Intrusions] LOGS: GIAC GCIA Version 3.4 Practical Detect (Kam Ng)
kam ng *<my_email_address>Sun Apr 18 15:06:42 UTC 2004*

Second Post: [Intrusions] LOGS: GIAC GCIA Version 3.4 Practical Detect (Kam Ng) Kam Ng *<my_email_address>Mon Apr 26 14:55:38 UTC 2004*

Network Detect #2 : SHELLCODE x86 0xEB0C NOOP - ftp shellcode exploit

[**] [1:1424:6] SHELLCODE x86 0xEB0C NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
11/02/02-05:09:06.526507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x23E
195.232.55.6:1701 -> 207.166.87.42:21 TCP TTL:45 TOS:0x0 ID:55450 IpLen:20 DgmLen:560 DF
AP Seq: 0x82340FD5 Ack: 0x4333A866 Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 1040178 3948516

Source of Trace
The raw log was download from http://isc.sans.org/logs/raw/2002.10.2

All the analysis was done on a laptop running Fedora Linux (2.6.3 kernel). In order to aid the analysis of the detect, I wrote several scripts. To use these scripts, I exported the raw log file to text format file first by running the following commands.

- tcpdump -r 2002.10.2 -ne >> log.txt

The 'scanlog' [20] script was used to summarize the traffic flow and to guess network devices in the system. Network devices like router, firewall and gateway will normally have more than one IP address associated with their MAC address. The oui.txt from IEEE (http://standards.ieee.org/regauth/oui/oui.txt) was also used as another indicator for network equipment. If the NIC belongs to Cisco, it's higher chance that it's a network device.

Network traffic summary
The script produced a very long list of IP address, too long to be put here. I ran it based on host and class A, B and C subnet type to guess the network

infrastructure. I concluded that Class B subnet 207.166.0.0/16 was hiding behind a Cisco device with MAC 00:00:0c:04:b2:33 on its outside interface. I took 207.166 class B subnet as the home subnet to protect.

Sample traffic out from this home class B subnet:

00:00:0c:04:b2:33(CISCOSYSTEMS,INC.)|207.166|-->|00:03:e3:d9:26:c0 (CiscoSystems,Inc.)<N-D>|12
00:00:0c:04:b2:33(CISCOSYSTEMS,INC.)|207.166|-->|00:03:e3:d9:26:c0 (CiscoSystems,Inc.)<N-D>|128
00:00:0c:04:b2:33(CISCOSYSTEMS,INC.)|207.166|-->|00:03:e3:d9:26:c0 (CiscoSystems,Inc.)<N-D>|129

Sample traffic into the home class B subnet:

00:03:e3:d9:26:c0(CiscoSystems,Inc.)<N-D>|128.167|-->|00:00:0c:04:b2:33 (CISCOSYSTEMS,INC.)|207.166
00:03:e3:d9:26:c0(CiscoSystems,Inc.)<N-D>|131.230|-->|00:00:0c:04:b2:33 (CISCOSYSTEMS,INC.)|207.166
00:03:e3:d9:26:c0(CiscoSystems,Inc.)<N-D>|134.53|-->|00:00:0c:04:b2:33 (CISCOSYSTEMS,INC.)|207.166

All Mac and IP inventory list
The script produced a very long list to be included here. It's absence should not hinder the understanding of this detect.

MAC/IP duplicate use statistics (The number in the third column separated with | tells how many IP were used with the MAC address. More than one will normally indicates a network devices such as router, firewall or gate.):

multi ip|src mac|2|00:00:0c:04:b2:33(CISCOSYSTEMS,INC.)
multi ip|src mac|45|00:03:e3:d9:26:c0(CiscoSystems,Inc.)
multi ip|dst mac|984|00:00:0c:04:b2:33(CISCOSYSTEMS,INC.)
multi ip|dst mac|1134|00:03:e3:d9:26:c0(CiscoSystems,Inc.)

From the data collected above, I believe that the traffic was monitored between a Cisco firewall/border router and a uplink/ISP (Internet Service Provider) Cisco router/switch.

```
                ( Internet )
                     |
[ Cisco router/switch 00:03:e3:d9:26:c0]          <- ISP
                     |
                     | <--- [IDS]
                     |
[ Cisco firewall/border router 00:00:0c:04:b2:33]  <- Corporate border
                     |
[Inside Class B subnet 207.166.0.0/16 ]           <- Internal network
```

Script scan_allow2 [22] failed to check the allowed inbound and outbound services. It's because the log only contains traffic that trigger snort alert according to the log's website. Quoted from the site's readme: ' The log files are the result of a Snort instance running in binary logging mode. This means that only the packets that violate the rule set will appear in the log.'

Detect was generated by
Snort Version 2.1.2 (Build 25) was used to generate the alert. The following command line was used:
snort -r 2002.10.2 -h 207.166.0.0/16 -l log -k none -c snort.conf -yeX

Switch -k was used because the readme in the incident.org raw log folder mentions that the checksum has been modified.

The snort.conf has all the rules selected and the flow-portscan preprocessor turned on.

Snort created directory 131.230.217.56, 134.53.212.120, 140.128.251.21,72.184.145.241, 192.6.234.9, 195.232.55.6, 198.150.73.5, 205.188.135.151, 213.99.232.219, 216.77.216.154, 24.159.255.58, 24.165.202.171, 255.255.255.255, 62.103.210.66, 64.154.80.44, 64.154.80.47, 64.154.80.51, 65.65.24.94, 67.36.84.5, 68.3.48.37, 68.39.106.20, 68.85.156.58, 80.6.58.50 and the alert file.

Script 'scanalertip' [23] was then run against the 'alert' file to summarize the attack signature and IP address information. It produces 1012 items which is too long to be listed here. Instead I just list the sample of he attack I picked for analysis.

1 [**] SHELLCODE x86 0xEB0C NOOP [**]|195.232.55.6:1701|->|207.166.87.42:21|TCP
1 [**] SHELLCODE x86 0xEB0C NOOP [**]|195.232.55.6:1710|->|207.166.87.40:21|TCP
1 [**] SHELLCODE x86 0xEB0C NOOP [**]|195.232.55.6:1737|->|207.166.87.41:21|TCP

This alert was generated by the following snort rule.

alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 0xEB0C NOOP"; content:"|EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C|"; classtype:shellcode-detect; sid:1424; rev:6;)

This is a simple rule that checks for a unique repeated pattern of EB0C.

Actual captured traffic with some payload data to illustrate the rule triggering point:

05:09:06.526507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4, length 574: IP (tos 0x0, ttl 45, id 55450, offset 0,
flags [DF], length: 560, bad cksum 9bb8 (->516e)!) 195.232.55.6.l2tp > 207.166.87.42.ftp: P [bad tcp cksum 7135 (->25e2)!] 2184450005:2184450513 (508) ack 1127458918 win 5840 <nop,nop,timestamp 1040178 3948516>

```
0x0000   4500 0230 d89a 4000 2d06 9bb8 c3e8 3706        E..0..@.-.....7.
0x0010   cfa6 572a 06a5 0015 8234 0fd5 4333 a866        ..W*.....4..C3.f
0x0020   8018 16d0 7135 0000 0101 080a 000f df32        ....q5.........2
0x0030   003c 3fe4 4357 4420 3030 3030 3030 3030        <?.CWD.00000000
0x0040   3030 3030 3030 3030 3030 3030 3030 3030        0000000000000000
<snip>
0x0120   3030 3030 3030 3030 3030 3030 3030 3030        0000000000000000
0x0130   3030 3030 3030 3030 f0fc 4031 0708 985f        00000000..@1..._
0x0140   0808[eb0c eb0c eb0c eb0c eb0c eb0c   eb0c      ................
0x0150   eb0c] eb0c eb0c eb0c eb0c eb0c eb0c eb0c       ................
<snip>
0x01e0   eb0c eb0c eb0c eb0c 9090 9090 9090 9090        ................
0x01f0   9090 9090 31db 43b8 0b74 510b 2d01 0101        ...1.C..tQ.-...
0x0200   0150 89e1 6a04 5889 c2cd 80eb 0e31 dbf7        P..j.X......1..
0x0210   e3fe ca59 6a03 58cd 80eb 05e8 ed0a ca59        ..Yj.X........Y
0x0220   6a03 58cd 80eb 05e8 edff ffff ffff ff0a        j.X.............
```

NOTE: [eb0c .. eb0c] marked above meets the matching criteria.

Probability the source address was spoofed
What's the intent of this exploit? It's mainly to gain shell access on the victim system. Because of this, it's very unlikely that the source is spoofed.

I also looked into the tcp/ip packets to see sign of spoofed source IP. I ran the command:

tcpdump -r 2002.10.2 -n -vv -S '(src host 195.232.55.6 and dst net 207.166.87)

or (src net 207.166.87 and dst host 195.232.55.6)' | cut -d " " -f 1,5,6,7,8,9,14,15,20,21,22,29,30,31 | more

-r: read from file 2002.10.2
-n: don't resolve IP to name
-vv: more verbose
-S: absolute sequence number

Result:

```
05:09:06.526507 ttl  45, id 55450, length: 560, 195.232.55.6.l2tp >
207.166.87.42.ftp: 2184450005:2184450513(508) ack 1127458918
05:09:06.976507 ttl  45, id 55451, length: 68, 195.232.55.6.l2tp >
207.166.87.42.ftp: 2184450513:2184450529(16) ack 1127459439
05:09:09.716507 ttl  45, id 55459, length: 59, 195.232.55.6.l2tp >
207.166.87.42.ftp: 2184450617:2184450624(7) ack
112745975005:10:00.686507 ttl  45, id 37808, length: 560, 195.232.55.6.1710 >
207.166.87.40.ftp: 2255257131:2255257639(508) ack 1173953630
05:10:01.146507 ttl  45, id 37809, length: 68, 195.232.55.6.1710 >
207.166.87.40.ftp: 2255257639:2255257655(16) ack 1173954151
05:10:03.846507 ttl  45, id 37817, length: 59, 195.232.55.6.1710 >
207.166.87.40.ftp: 2255257743:2255257750(7) ack
117395446205:10:51.706507 ttl  45, id 11897, length: 560, 195.232.55.6.1737 >
207.166.87.41.ftp: 2305939120:2305939628(508) ack 1223541630
05:10:52.156507 ttl  45, id 11898, length: 68, 195.232.55.6.1737 >
207.166.87.41.ftp: 2305939628:2305939644(16) ack 1223542151
05:10:54.846507 ttl  45, id 11906, length: 59, 195.232.55.6.1737 >
207.166.87.41.ftp: 2305939732:2305939739(7) ack
122354246205:11:41.886507 ttl  45, id 29061, length: 560, 195.232.55.6.1756 >
207.166.87.60.ftp: 2365219651:2365220159(508) ack 1273684817
05:11:42.356507 ttl  45, id 29064, length: 68, 195.232.55.6.1756 >
207.166.87.60.ftp: 2365220159:2365220175(16) ack 1273685338
05:11:45.016507 ttl  45, id 29072, length: 59, 195.232.55.6.1756 >
207.166.87.60.ftp: 2365220263:2365220270(7) ack
127368564905:12:32.626507 ttl  45, id 14521, length: 560, 195.232.55.6.1763 >
207.166.87.58.ftp: 2407932020:2407932528(508) ack 1330230075
05:12:33.056507 ttl  45, id 14522, length: 68, 195.232.55.6.1763 >
207.166.87.58.ftp: 2407932528:2407932544(16) ack 1330230596
05:12:36.046507 ttl  45, id 14530, length: 59, 195.232.55.6.1763 >
207.166.87.58.ftp: 2407932632:2407932639(7) ack 133023090
```

I see that:

1. The TTL is the same for all the traffic. This very likely indicates that these captured exploits were coming from the same physical location.
2. The packet length pattern was consistent for all the exploit against different hosts. This is very likely that the target systems were under the same attacks.
3. The sync seq number could have been more random. This could mean

crafted packets but could just be a system signature.
4. I would expect the IP id should be linearly increased but not within this short period of time unless the system was very busy with its network or it's a newer OS that uses random IP id.

There's no strong indications that these are crafted packets. So based on all these observations, I still believe that the source is not spoofed.

Description of Attack:
I ran the following command to summarize the network connections between attacker and target systems:

tcpdump -r 2002.10.2 -n -vv '(src host 195.232.55.6 and dst net 207.166.87) or (src net 207.166.87 and dst host 195.232.55.6)' | cut -d " " -f 20,21,22 | sort -u

Result:
195.232.55.6.1710 > 207.166.87.40.ftp:
195.232.55.6.1737 > 207.166.87.41.ftp:
195.232.55.6.1756 > 207.166.87.60.ftp:
195.232.55.6.1763 > 207.166.87.58.ftp:
195.232.55.6.l2tp > 207.166.87.42.ftp:

The attacker was at system with IP 195.232.55.6 sitting on the Internet side and targeted five ftp systems located inside the class B 207.166.0.0/16 network. The attack targeted x86/linux wu_ftpd remote root exploit based on a memory handling problem of the globbling codes in the ftp daemon that allows user to run codes of their choice under certain operating conditions.

I believe that exploit code was similar to exploit code called 7350wurm and its variants. It was written to target wu-ftp server but other ftp serves are also found to be vulnerable as Stephen's GCIH paper points out. [3.6].This could also be coming from a network scanner.

The code can be found at: http://www.packetstormsecurity.org/0205-exploits/7350wurm.c

Attack Mechanism
Snort only alerts 'SHELLCODE x86 0xEB0C NOOP' for these systems in this raw log. However, you can see from below that this signature is only one part of the exploit that sent the necessary shell code into the buffer.

According to the report at [3.3], in order to exploit this, the attacker must have ftp access to the system first, either as a valid ftp user or anonymous users. This is evident that it's the case as the attacker was able to issue a ftp CWD (change directory) command on the system. However, this portion of traffic was missing

from the raw log. According to the raw log readme, only packets that violate the rule set will appear in the log.

In order to understand the actual traffic for comparison purpose, I compiled the download 7350wurm C code and watched the traffic using tcpdump when it attacked another PC running ftp daemon on my home network.

To do all these, I ran:
1. 'gcc 7350wurm.c -o ftp.exp' to generate the executable code.
2. 'tcpdump -i eth0 -n -xxxV -s 1024 'src host <attacker> or src host <target IP>' | tee -a >> wuftp.dump' to capture the actual traffic. With the utility 'tee', I can watch the packets and send them to a file at the same time.
3. run './ftp.exp -v -a -u <user> -p <password> -d <target IP>'

From the tcpdump output, the exploit flowed as showed below:
1. Attacker log on to the ftp server successfully using the user id and password supplied.
2. It issued a number of RNFR ././. to fill up the wuftp heap space which did not free up properly.
3. Then it sent a CWD command with a long argument of zeros, others data, the short shell code and its address pointer. The data sent was very similar even not 100% to the data in the raw log of this detect.
4. Then it sent more packets containing the CWD command with different combination of ~, { and ./. The real killer was the last CWD ~{ command sent, which ran the exploit code in the previous packet sent. These packets were seen in the raw log also.
5. Once this exploit is successful, a linux shell code was sent to run setreuid(0,0) and a shell was created with root privilege.
6. When all above were execute successfully, the attacker owned the system.

Correlation

It took some time at the beginning to found detail information about this alert until I googled '\x31\xdb\x43\xb8' which is part of the exploit code.

The information pertaining to this alert are posted at the websites below.

[3.1] CERT. “Advisory CA-2001-33 Multiple Vulnerabilities in WU-FTPD” 12 Feb 2003
URL: http://www.cert.org/advisories/CA-2001-33.html
[3.2] 73050wurm source code
URL: http://www.packetstormsecurity.org/0205-exploits/7350wurm.c
[3.3] Corelabs. “Vulnerability Report for WU-FTPD Server” 28 Nov 2001.
http://www1.corest.com/common/showdoc.php?idxseccion=10&idx=172
[3.4] Power, Matt. “FTP exploit discussion” 20 Apr 2001.
URL: http://archives.neohapsis.com/archives/vuln-dev/2001-q2/0311.html
[3.5] Martynox, Evgueni. “GCIA Practical Assignment V3.2”
URL: http://www.giac.org/practical/GCIA/Evgueni_Martynov_GCIA.doc
[3.6] Hall, Stephen. “A variant of the WU-FTPD File Globbing Heap Corruption

Vulnerability" 17 Nov 2003.
URL: http://www.giac.org/practical/GCIH/Stephen_Hall_GCIH.pdf

Evidence of Active targeting
It is evident that the attacker at 195.232.55.6 was actively targeting four ftp servers located on subnet 207.166.0.0/16.

Unfortunately, there's no traffic in the raw log that I could use to check if the attack was successful. There's no returning traffic to the attacking system. I ran 'tcpdump -r 2002.10.2 -ne 'dst host 195.232.55.6" and nothing returned.

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

4 = (4 + 5) - (2 + 3)

Criticality = 4. As I could see the actual glob exploit traffic from the raw log, the attacker must have gained ftp access already. This tells me that they were actually ftp servers. There were five of them. They should belong to a business organization. It's very likely they are part of the business infrastructure.

Lethality = 5. This vulnerability has the potential to allow attacker to run code of their own choice. There's a potential to gin root access.

System Countermeasures = 2. Obviously, the system was missing patches to allow the exploit.

Network Countermeasures = 3. The organization has a business to run. I assume that the firewall/border was configured properly to restrict access to their servers. This vulnerability is at application layer.

A score of severity of 4 is a bad security situation that immediate action must be taken to mitigate the risks.

Defensive Recommendation
We have to address the following:
1. Make sure the network control only allow opening for services with real business need?
2. The system is not vulnerable to current exploits at least for the services open to any untrusted zone. Internet is one example.

For point #1:
1. The network security should be reviewed to make sure proper control is in place.

2. The Internet point of presence should be audit on a regular basis.
3. All these should not be a one time task but on-going.

For point #2:
1. If the system was compromised, the current system security alert monitoring and patching process must be reviewed to understand why this time the exploit was successful and improve the process based on the lesson learned. Also, the system configuration has to be reviewed to make sure proper security measures are in place to defend the ftp service from attackers.
2. Are we offering more than necessary ftp services to the outside. Do we need to offer anonymous ftp? If not, shut them down. The very basic and effective security principle is not to offer unnecessary services.

If the above recommendations can be carried out, I will give system countermeasures a four.

With these countermeasure improvement in place, we can re-calculate the severity.

Severity = (4 + 5) - (4 + 3) = 2, an improvement over 4. But still this indicates that system is vulnerable to unknown exploit even the administrators are vigilant enough on maintaining the system patches up to date.

This is a risk to run their business. This has to be communicate to management. And a good incident handling process must be setup if not in place yet.

Multiple choice test question
What is the RNFR command in ftp:
a. Specify the old file name to be renamed.
b. Remove non-existing file resources.
c. Read the number of the file record.
d. None of the above

Answer: a.

Reference: RFC959

Verify the Snort Running On My System
I was missing some of the ftp exploit related alerts that I expected to see during this analysis. What is the problem? I determined to check the situation by using snort on my system to alert exploits generated by running ftp.exp.

I ran the following commands for this trouble shooting:
1. Snort: snort -i eth0 -h 192.168.2.116/32 -l log -c snort.conf -yeX in which 192.168.2.116 is the target system running with wu-ftp server.

2. ftp exploit: ./ftp.exp -a -v -u kam -p <password> -d 192.168.2.116

I ran 'scanalertip' to summarize snort alerts generated:

2 [**] FTP CWD ~ attempt [**]|192.168.2.146:33101|->|192.168.2.116:21|TCP
75 [**] FTP RNFR ./. attempt [**]|192.168.2.146:33101|->|192.168.2.116:21|
TCP
1 [**] SHELLCODE x86 0xEB0C NOOP [**]|192.168.2.116:21|->|
192.168.2.146:33101|TCP
1 [**] SHELLCODE x86 0xEB0C NOOP [**]|192.168.2.146:33101|->|
192.168.2.116:21|TCP

This proves that snort running on my system is working as expected. So where is the problem? I took a look at the 'FTP CWD ~ attempt' rule in the ftp.rules file.

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP CWD ~ attempt"; flow:to_server,established; content:"CWD"; pcre:"/^CWD\s+~/smi"; reference:cve,CAN-2001-0421; reference:bugtraq,2601; reference:bugtraq,9215; classtype:denial-of-service; sid:1672; rev:7;)

The only thing I could think of that may cause this rule to fail is missing the corresponding state packets in the raw log for the flow rule to decode. This is very likely the case as I checked that lot of the packets pertaining to this attack were missing from this raw log.

Network Detect #3 : BAD-TRAFFIC ip reserved bit set
[**] [1:523:4] BAD-TRAFFIC ip reserved bit set [**]
[Classification: Misc activity] [Priority: 3]
06/15-21:25:49.004488 192.1.1.188 -> 46.5.226.187
TCP TTL:239 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset: 0x0864   Frag Size: 0x0014

Source of Trace
The raw log was download from http://isc.sans.org/logs/Raw/2002.5.16. All the analysis was done on a laptop running Fedora Linux (2.6.3 kernel). In order to aid the analysis of the detect, I wrote several scripts . To use these scripts, I exported the raw log file to text format file first by running the following commands.

- tcpdump -r 2002.5.16 -ne >> log.txt

The 'scanlog' [20] script was used to summarize the traffic flow and to guess network devices in the system. Network devices like router, firewall and gateway will normally have more than one IP address associated with their MAC address. The oui.txt from IEEE (http://standards.ieee.org/regauth/oui/oui.txt) was also

used as another indicator for network equipment. If the NIC belongs to Cisco, it's higher chance that it's a network device.

Network traffic summary
The script produced a very long list of IP address, too long to be put here. I ran it based on host and class A, B and C subnet type to guess the network infrastructure. I concluded that: Class B subnet 46.5.0.0/16 was hiding behind a Cisco device with MAC 00:00:0c:04:b2:33 on its outside interface. I took 46.5.0.0 class B subnet as the home subnet to protect.

Sample traffic from this home class B subnet:

00:00:0c:04:b2:33(CISCOSYSTEMS,INC.)<N-D>|46.5.180|-->|00:03:e3:d9:26:c0(CiscoSystems,Inc.)<N-D>|152.163.209
00:00:0c:04:b2:33(CISCOSYSTEMS,INC.)<N-D>|46.5.180|-->|00:03:e3:d9:26:c0(CiscoSystems,Inc.)<N-D>|204.253.104
00:00:0c:04:b2:33(CISCOSYSTEMS,INC.)<N-D>|46.5.180|-->|00:03:e3:d9:26:c0(CiscoSystems,Inc.)<N-D>|208.184.29

Sample traffic into the home class B subnet:

00:03:e3:d9:26:c0(CiscoSystems,Inc.)<N-D>|12.108.43|-->|00:00:0c:04:b2:33(CISCOSYSTEMS,INC.)<N-D>|46.5.15
00:03:e3:d9:26:c0(CiscoSystems,Inc.)<N-D>|12.108.43|-->|00:00:0c:04:b2:33(CISCOSYSTEMS,INC.)<N-D>|46.5.64
00:03:e3:d9:26:c0(CiscoSystems,Inc.)<N-D>|12.108.43|-->|00:00:0c:04:b2:33(CISCOSYSTEMS,INC.)<N-D>|46.5.7

All Mac and IP inventory list
The script produced a very long list to be included here. It's absence should not hinder the understanding of this detect.

MAC/IP duplicate use statistics (The number in the third column separated with | tells how many IP were used with the MAC address. More than one will normally indicates a network devices such as router, firewall or gate.):

multi ip|src mac|35|00:03:e3:d9:26:c0(CiscoSystems,Inc.)
multi ip|dst mac|109|00:00:0c:04:b2:33(CISCOSYSTEMS,INC.)
multi ip|dst mac|11|00:03:e3:d9:26:c0(CiscoSystems,Inc.)

It's interesting to observe that MAC 00:00:0c:04:b2:33 was used as destination only. This is very rare for such a network. The problem could be coming from my script, the log is missing something or it's the way it is.

To verify this, I manually check this by running the following command.

tcpdump -r 2002.5.16 -ne 'ether src 00:00:0c:04:b2:33' | cut -d " " -f 10 | cut -d "." -f 1-4 | uniq -c

The result was : 577 count of IP 46.5.180.250 using this Mac as source, no other IP. Only one IP had used this MAC as source. I could safely say that my script was working as expected.

From the data collected above, I believe that the traffic was monitored between a Cisco firewall/border router and a uplink/ISP (Internet Service Provider) Cisco router/switch.

```
                ( Internet )
                      |
[ Cisco router/switch 00:03:e3:d9:26:c0]          <-- ISP
                      |
                      | <--- [IDS]
                      |
[ Cisco firewall/border router 00:00:0c:04:b2:33]  <-- Corporate border
                      |
[Inside Class B subnet 46.5.0.0/16 ]              <-- Internal network
```

Detect was generated by
Snort Version 2.1.2 (Build 25) was used to generate the alert. The following command line was used:

snort -r 2002.5.16 -h 46.5.0.0/16 -l log -k none -c snort.conf -yeX

Switch -k was used because the readme in the incident.org raw log readme mentions that the checksum has been modified.

The snort.conf has all the rules selected and the flow-portscan preprocessor turned on.

Snort created directories 12.108.43.5, 159.226.208.40, 163.22.229.253, 163.23.190.2, 163.23.190.34, 192.1.1.188, 194.108.153.205, 194.78.59.253, 200.184.104.200, 202.96.52.99, 203.197.102.29, 209.236.203.101, 11.152.3.40 212.88.236.2, 218.96.62.2, 255.255.255.255, 61.218.166.106, 61.218.166.98, 61.221.99.242, 64.12.151.56, 64.154.80.50, 64.154.80.51, 64.94.89.210 and the alert file.

Under 192.1.1.188, there's only one file, IP_FRAG

Script 'scanalertip' [23] was then run against the 'alert' file to summarize the attack signature and IP address information. The output is listed immediately below. The leftmost number is the signature hit count.

1 [**] BAD-TRAFFIC ip reserved bit set [**]|06/15-21:25:49.004488|
192.1.1.188|->|46.5.226.187
1 [**] BAD-TRAFFIC ip reserved bit set [**]|06/15-21:32:20.144488|
192.1.1.188|->|46.5.20.200
1 [**] BAD-TRAFFIC ip reserved bit set [**]|06/16-02:11:01.594488|
192.1.1.188|->|46.5.132.21

This alert was generated by the following snort rule.

alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"BAD-TRAFFIC ip reserved bit set"; fragbits:R; sid:523; classtype:misc-activity; rev:4;)

This is a relatively simple rule that detect IP traffic with the reserved flag bit located at bit 8 of byte 6 of the IP header, counting from 0.

21:25:49.004488 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4, length 60: IP (tos 0x0, ttl 239, id 0, offset 17184, flags [rsvd], length: 40, bad cksum 76f3 (->70ed)!) 192.1.1.188 > 46.5.226.187: tcp
0x0000   4500 0028 0000 [8]864 ef06 76f3 c001 01bc   E..(...d..v.....
0x0010   2e05 e2bb 0723 0050 1275 894e 1275 894e   .....#.P.u.N.u.N
0x0020   0004 0000 f46e 0000 0000 0000 0000   .....n.....

NOTE: [8] marked above meets the matching criteria. This where the IP flag field located.

Probability the source address was spoofed
A visit to samspade returned the following network information for 192.1.188:

OrgName: BBN Communications
OrgID: BBNP
StateProv: MA
Country: US
NetRange: 192.1.0.0 - 192.1.255.255
CIDR: 192.1.0.0/16

This is within the Internet route-able IP address range.

The traffic generated was TCP but there's not enough information from this raw log that I could inspect the traffic state to determine if the source IP was spoofed.

What does Snort site say about this rule at http://www.snort.org/snort-db/sid.html?sid=523. Quoted:'Under normal circumstances IP packets do not use the reserved bit. This may be an indicator of the use of the reserved bit by a

malicious user to instigate covert channel communications. An indicator of unauthorized network use, reconnaissance activity or system compromise. These rules may also generate an event due to improperly configured network devices.'

For most of the covert channel, reconnaissance activity, system compromise and the result of mis-configured network devices, the source IP addresses are not spoofed. There are stealthy scanning technique involves a silent host but it's only good at checking if a host or the service is up but not finger-printing.

We all know that some of the security tools such nmap use specially crafted packets to finger-print systems. Most cases, the source IP addresses are not spoofed.

Based on all I could get out of the raw log, Internet and reading research, I believe it's very unlikely the source IP address was spoofed.

Description of Attack
The attack was originated from system with IP 192.1.1.188. I ran the following command trying to get all the traffic related to the attacker.

tcpdump -r 2002.5.16 -n 'host 192.1.1.188'.
Result:
21:25:49.004488 IP 192.1.1.188 > 46.5.226.187: tcp
21:32:20.144488 IP 192.1.1.188 > 46.5.20.200: tcp
02:11:01.594488 IP 192.1.1.188 > 46.5.132.21: tcp
05:10:30.204488 IP 192.1.1.188 > 46.5.72.213: tcp
07:14:48.704488 IP 192.1.1.188 > 46.5.130.136: tcp
14:12:37.084488 IP 192.1.1.188 > 46.5.94.235: tcp
14:43:14.304488 IP 192.1.1.188 > 46.5.178.85: tcp
15:44:30.484488 IP 192.1.1.188 > 46.5.51.201: tcp
17:36:39.194488 IP 192.1.1.188 > 46.5.109.103: tcp

The normal attributes that comes with a TCP packet are missing, e.g. the sequence number, the flags, the windows size and so on.

Let see why it's the case by reviewing the actual data in one of the packets.

21:25:49.004488 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4, length 60: IP (tos 0x0, ttl 239, id 0, offset 17184, flags [rsvd], length: 40, bad cksum 76f3 (->70ed)!) 192.1.1.188 > 46.5.226.187: tcp
0x0000 4500 0028 0000 8864 ef06 76f3 c001 01bc
0x0010 2e05 e2bb 0723 0050 1275 894e 1275 894e
0x0020 0004 0000 f46e 0000 0000 0000 0000

Bytes 4 to 5 (=0000) is the IP ID field. It's 0.

Bytes 6 to 7 (=8864) is the flags and fragment offset field. They are broken down below for easy read.

Byte 6, bit 7 = reserved bit. Normally set to 0 (zero) but was 1.
Byte 6, bit 6 = Don't Fragment bit set to zero.
Byte 6, bit 5 = More fragment bit set to zero.
Byte 6, bit[4-0] + Byte 7, bit[7-0] = IP fragment offset. Set to
0x864 which equals 0x864 x 8 bytes = 17184 decimal.

Byte 9 (=06) indicating it's a TCP protocol

Putting all these observations together:
1. It's TCP packets.
2. It's fragmented packet and the fragment offset is non-zero. Therefore the normal TCP attributes such as sequence number and TCP flags were missing.
3. It's the last packet as the MF flag was not set.

I ran the following command to list some of the attributes for all the detected traffic to look for similarity and difference.

```
tcpdump -r 2002.5.16 -ne -vvv 'host 192.1.1.188' | cut -d " " -f 7-21,26-29

length 60: IP (tos 0x0, ttl 239, id 0, offset 17184, flags [rsvd], length: 40,
192.1.1.188 > 46.5.226.187: tcp
length 60: IP (tos 0x0, ttl 239, id 0, offset 17184, flags [rsvd], length: 40,
192.1.1.188 > 46.5.20.200: tcp
length 60: IP (tos 0x0, ttl 239, id 0, offset 17184, flags [rsvd], length: 40,
192.1.1.188 > 46.5.132.21: tcp
length 60: IP (tos 0x0, ttl 239, id 0, offset 17184, flags [rsvd], length: 40,
192.1.1.188 > 46.5.72.213: tcp
length 60: IP (tos 0x0, ttl 239, id 0, offset 17184, flags [rsvd], length: 40,
192.1.1.188 > 46.5.130.136: tcp
length 60: IP (tos 0x0, ttl 239, id 0, offset 17184, flags [rsvd], length: 40,
192.1.1.188 > 46.5.94.235: tcp
length 60: IP (tos 0x0, ttl 239, id 0, offset 17184, flags [rsvd], length: 40,
192.1.1.188 > 46.5.178.85: tcp
length 60: IP (tos 0x0, ttl 239, id 0, offset 17184, flags [rsvd], length: 40,
192.1.1.188 > 46.5.51.201: tcp
length 60: IP (tos 0x0, ttl 239, id 0, offset 17184, flags [rsvd], length: 40,
192.1.1.188 > 46.5.109.103: tcp
```

These packet are having the following common characteristics:
1. Total packet length = 60 bytes
2. TOS = 0x0
3. TTL = 239
4. IP = 0
5. IP fragment offset = 17184

6. IP reserved flag bit set.
7. TCP protocol.

It's the IP ID = 0 makes me to believe that this is a crafted packet. You may be having 1 out of 65535 chance to run with IP ID = 0 but not everyone of them.

What can the attacker gain from sending this packet? Gaining insight in the target system by running system reconnaissance and finger printing activities, I believe.

Attack Mechanism
I believe these packets are part of system reconnaissance or finger printing activities. However, I don't believe sending these packets alone would be able to solicit any response from the target systems.

In order to verify this point I just made, I used tool to craft similar packets to see if I could get any response from Windows98 and Redhat 9 system which I have at home. I used hping2 with a minor modification as it does not have the option to modify the IP 'reserved bit'.

I did not have the time to spend to understand the whole hping program, I did a dirty trick to disable the right shifting of the fragment offset parameter and then adjusted the offset parameter in the command line to trick the program to send the packets with 'reserved bit' set.

Below is the modification in the sendip.c:

```
/*    ip->frag_off    |= htons(fragoff >> 3); shift three flags bit */
      ip->frag_off    |= htons(fragoff); /* Kam - dirty trick to set reserved bit */
```

Then I ran the following on two different terminal sessions on my linux laptop:
1. hping2 192.168.2.100 --id 0 --ttl 239 --tos 0 --fragoff 34916 -c 5 -n

- 192.168.2.100 is the target systems, Linux and Windows98.
- id (IP ID) = 0 but this was not zero at run time. But this should not matter.
- tos (Type of service) = 0
- ttl (Time to Live) = 239
- fragoff (IP fragment offset) = 34916 = 0x8864. The first digit 8 set the reserved bit but not the other fragment flag bit. I used the value of byte 6 and 7 found in the packets that triggered the 'ip reserved bit' rule.
- c (count) = 5, run 5 times
- n (do not resolve name)

Output:
HPING 192.168.2.116 (eth0 192.168.2.116): NO FLAGS are set, 40 headers + 0 data bytes

--- 192.168.2.116 hping statistic ---
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

Capture the hping2 traffic using tcpdump -i eth0 -vvvX -n

Output (one sample, others were the same):
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
20:08:35.740653 IP (tos 0x0, ttl 239, id 63990, offset 17184, flags [rsvd], length: 40) 192.168.2.146 > 192.168.2.100: tcp
0x0000: 4500 0028 f9f6 8864 ef06 c32d c0a8 0292 E..(...d...-....
0x0010: c0a8 0264 0586 0000 55e2 40c1 558e 4803 ...d....U.@.U.H.
0x0020: 5000 0200 ede2 0000   P.......

There's no response from the target systems, both Windows98 and Linux systems.

This test makes me to believe that this stimulus will not generated response from system. Therefore, I tends to believe that the raw log is missing some important information about the pre-scan and possibly post-scanning activities.

If I am allowed to take a wild guess, I believe that someone must have send the first TCP segment to initiate the handshake and then sent this packet to finger print the system.

Correlation
There were some information on the Internet but this signature has been discussed a lot. Some discussion were posted at the following websites:
1. Roesch, Martin. “Snort discussion thread”. 11 Oct 2001.
URL: http://archives.neohapsis.com/archives/snort/2001-10/0357.html
2. Granier, Thomas. “GCIA version 3.3 Practical Detect – Reserved bit set” 27 Nov 2002
URL: http://www.dshield.org/pipermail/intrusions/2002-November/006015.php
3. Macbeth, Soren. “GCIA version 3.3 Practical Detect”.8 Oct 2002.
URL: http://www.dshield.org/pipermail/intrusions/2002-October/005553.php
4. Snort. Rule search.
URL: http://www.snort.org/snort-db/sid.html?sid=523 – snort explanation
5. Granier, Brian. “Intrusion Analysis and LaBra Sentry” 21 Oct 2002.
URL: http://www.giac.org/practical/GCIA/Brian_Granier_GCIA.pdf

Evidence of Active targeting
The network activities generated by this exploit is summarized below:
192.1.1.188|->|46.5.226.187
192.1.1.188|->|46.5.20.200

192.1.1.188|->|46.5.132.21
192.1.1.188|->|46.5.72.213
192.1.1.188|->|46.5.130.136
192.1.1.188|->|46.5.94.235
192.1.1.188|->|46.5.178.85
192.1.1.188|->|46.5.51.201
192.1.1.188|->|46.5.109.103

Judging from the target system IP address pattern, I believe these systems were not being actively targeted. Instead, this was more to me of a system reconnaissance and finger printing Internet activities.

Of course, someone can argue that the attacker could be actively targeting the whole organization. This may be the case but I don't believe anyone is having solid information to back up to their claims on either sides.

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

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

Criticality = 3. I don't think the raw log contains information about what kind of system they were. I gives it a three to be on the safe side.

Lethality = 2. I believe this is just a system reconnaissance and finger printing activities. But this could turn into a lethal situation if the attacker could locate vulnerabilities to penetrate.

System Countermeasures = 3. No information to determine this too. I give it a three assuming these systems are under normal management.

Network Countermeasures = 3. No information to determine this too. I give it a three assuming these systems are under normal management.

A score of severity of -1 is not a bad security situation. Immediate action is not necessary to be taken to mitigate risk.

Defensive Recommendation
To prevent the systems from future scan of this type:
1. If technical possible, the border router/firewall should be setup to block traffic that has the 'IP flag reserved bit' set. This could present problem if Explicit Congestion Notification is enabled. The situation has to be review with the network administrator.
2. The IDS should be setup to monitor this type of out-of-spec network traffic.

3. As this technique may be implemented in very slow stealthy scan, it's more effective to do trending report on out-of-spec traffic.

Multiple choice test question

Which packets below will trigger the 'BAD-TRAFFIC ip reserved bit set' snort rule:
a. 0x0000: 4500 0028 f9f6 8464 ef06 c32d c0a8 0292
b. 0x0000: 4500 0028 f6f9 2464 ef06 c32d c0a8 0292
c. 0x0000: 4500 0028 9f6f 2864 ef06 c32d c0a8 0292
d. 0x0000: 4500 0028 6f9f 0864 ef06 c32d c0a8 0292

Answer: a. It's the only packet contains a value in byte 6 that set the 'ip reserved bit'.

Part 3: Analyze This

Executive Summary
This report was prepared base on the findings from analyzing the logs supplied which covers period from 7th to 11th April, 2004. There are several areas that the University is recommended to consider to improve the computer security of the campus.

The IDS can be improved by further reducing the false positive. During the analysis, it was found that the system generate more false positive HTTP and other traffic than I believe it should be. This is a very preliminary observation. However, this seems to be a good opportunity to take a second look at the IDS system for improvement.

Through this exercise, it seems that the University is allowing quite wide open access between the campus network and the Internet. As we could understand that University in general wants to maintain close contacts with the outside to foster free exchange of information and encourage research cooperation. But the University should still run their computer infrastructure as securely as possible. Network in the campus should be properly segregated with appropriate security control so that the crown jewelry is sufficiently protected from attacks.

Seemingly, there are services that may not necessarily needed but allowed. This could have been setup for the students or are run by the students themselves. Some may just be abandon old systems. With these system running without appropriate administration, they are potential candidates of attack and then becoming attackers. This poses not just internal security concerns but liability of the University to the outside world. The University may still be paying for its maintenance such as computer room space, software license (very low of

course) and hardware maintenance fee, utility bill and etc. So the University should audit these systems to capture possible saving opportunities.

Running with an open network policy, the University should review the system maintenance policy to prioritize system patch management for systems at different security zones. Using a one size fits all approach will be too costly to the University which may end up with nothing done.

There are number of systems identified with possible security compromise. The University must investigate them and take appropriate actions to recover these systems. Also, the investigation results should be reviewed to prevent the same problems in the future.

The University should review the external attacker IP reported below to put a stop to the external attackers who may have been using the University computing resources to exploit others.

One of the University systems has been identified as an participant in an DDOS attack network. This poses danger to the University as the attacker can use the University's computing resources to launch a massive DDOS against the University itself or organizations on the outside. Either way, the image of the University will be damaged if that ever happened.

Source of Log files

These logs were download from http://isc.sans.org/logs/

Scan logs:	Alert logs:	Out-of-Spec logs:
scans.040407	alert.040407	oos_report_040403
scans.040408	alert.040408	oos_report_040404
scans.040409	alert.040409	oos_report_040405
scans.040410	alert.040410	oos_report_040406
scans.040411	alert.040411	oos_report_040407

For some reasons the time stamp of the OOS filename is not consistent with the actual time stamp in the log items. The time range of the files I am interested in is offset by 4 days later. That's why I used the files as listed to match time with the other logs. I believed it's more advantageous from analysis perspective to keep using logs from the same time period.

I did the following before using these logs for analysis:
1. Checked and cleaned up corrupted data. I encountered three logs with this problem.
2. Merged the individual type of logs into one single log in proper chronological

sequence. They are called alert.all (215MB) , oos.all (3.4 MB) and scans.all (1 GB). These logs cover events from 07 to 11 April, 2004.

3. Removed duplicated space in the field separator to ease log analysis.

Alert Analyzed

Before picking what alerts to analyze, I ran the following command to get the alerts statistics. Port scan activities were purposely filtered off (grep -v 'portscan') as most of them should be covered in the scan log analysis.

```
cat alert.all | sed 's/\[\*\*\]/|/g' | cut -d "|" -f 2 | grep -v 'portscan' | sort | uniq -c | sort -nr
```

Summary results (occurrence count Alert Message):

```
  28842  EXPLOIT x86 NOOP
  13011  MY.NET.30.3 activity
  12178  SMB Name Wildcard
  10667  High port 65535 tcp - possible Red Worm - traffic
  10212  MY.NET.30.4 activity
   8011  Tiny Fragments - Possible Hostile Activity
   3263  DDOS mstream handler to client
   1127  Null scan!
   1098  NMAP TCP ping!
   1082  Possible trojan server activity
    930  External RPC call
    638  SUNRPC highport access!
    512  Incomplete Packet Fragments Discarded
    309  TCP SRC and DST outside network
    245  High port 65535 udp - possible Red Worm - traffic
    210  ICMP SRC and DST outside network
    158  [UMBC NIDS] Internal MiMail alert
    147  [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.
    142  DDOS shaft client to handler
    108  [UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to
IRC
    100  FTP passwd attempt
     83  TCP SMTP Source Port traffic
     72  IRC evil - running XDCC
     66  EXPLOIT x86 setuid 0
     55  SMB C access
     47  [UMBC NIDS] External MiMail alert
     46  connect to 515 from outside
     33  EXPLOIT x86 setgid 0
     28  EXPLOIT x86 stealth noop
     25  [UMBC NIDS IRC Alert] Possible drone command detected.
     24  RFB - Possible WinVNC - 010708-1
```

22 FTP DoS ftpd globbing
17 [UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.
15 NIMDA - Attempt to execute cmd from campus host
14 TFTP - Internal UDP connection to external tftp server
14 Attempted Sun RPC high port access
13 SYN-FIN scan!
10 EXPLOIT NTPDX buffer overflow
8 EXPLOIT x86 NOPS

The 10 alerts in bold will be analyzed below. The reasons for their choice varies as shown in the table below.

Alert	**Reason**
EXPLOIT x86 NOOP	high volume alert
MY.NET.30.3 activity	high volume alert
SMB Name Wildcard	high volume alert
High port 65535 tcp - possible Red Worm - traffic	high volume alert
DDOS mstream handler to client	high volume alert and more critical than the previous two in the original list
Possible trojan server activity	high volume alert and more critical than the previous two scan types alerts in the original list.
External RPC call	high volume alert
FTP passwd attempt	something that catch my eye
IRC evil - running XDCC	IRC is a commonly used service
RFB - Possible WinVNC - 010708-1	could be used a system back door

Alert #1 - EXPLOIT x86 NOOP
Alert rule
I was not able to find this rule in my Version 2.1.2 (Build 25) rule base and the rule base in the snort website. So this may be a custom rule.

Even worse, google search returned two different set of rules. One is for UDP and one for TCP. They are very similar but the interpretation of the alerts based on two rules are quite different.

From http://www.security-labs.org/index.php3?page=408,
§§§§§
alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"EXPLOIT x86

NOOP";
content:"|9090 9090 9090 9090 9090 9090 9090 9090|";
reference:arachnids,181;)

From http://www.devking.com/networx/Intrusion%20Detection.doc,

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"EXPLOIT x86 NOOP"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; flags: A+; reference:arachnids,181;)

I have chosen to use the TCP version as port 80 was involved. It is very uncommon that UDP is used for port 80. UDP port 80 is very unlikely exploitable today, hackers are unlikely to us it.

This rule performs a simple detection of a consecutive occurrence of pattern hexadecimal data of 90 which is the no-operation (noop) code for Intel based processor. A long string of noop is a typical signature of a buffer overflow exploit of some kind against the target system, adding them into the exploit data packet to make the exploit somewhat more forgiving.

More analysis can be found at:
1. Oborn, David. “SANS GCIA Practical Assignment”.
URL: http://www.giac.org/practical/David_Oborn_GCIA.html#detect4
2. Larratt, Glenn. “GCIA Practical Assignment Version 3.0”
URL: http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html#x86ml

Analysis
First of all, I ran the following command to summarize this alert type.

cat alert.all | grep 'EXPLOIT x86' | sed 's/ \[**\] /|/g' | cut -d "|" -f 3 | gawk '{ split ($1, T, ":"); print T[1], $2, $3;}' | sort | uniq -c | sort -nr

Note: gawk in the command line above will remove the source port to reduce down the length of the list significant.

Sample Result:
```
  773 199.131.21.34 -> MY.NET.84.235:80
  767 199.131.21.34 -> MY.NET.84.236:80
  305 199.131.21.34 -> MY.NET.84.204:80
```

Note that the number on the rightmost column is the number of occurrence.

I put the output data into spreadsheet to pull statistics out of it. The result is shown in the table below.

Item	Source IP	Dest IP, MY.NET	Dest Port	Count
1	wide range	wide range	80	3163
2	wide range	190.102; 190.93; 190.95; 190.97	135	254
3	216.65.73.26	11.11; 11.3; 11.6; 11.7; 25.19; 29.13; 29.15; 70.5; 97.81	389	9
4	wide range	190.95; 190.97	445	14
5	wide range	wide range	1025	1210
6	wide range	wide range	5000	466
7	wide range	27.103; 66.32; 75.30; 75.6; 84.145	6129	16
8	wide range	wide range	others	162

Item 1:

1. This is very likely false positive of normal http traffic.
2. As this TCP data exchange, the involved systems must have completed the 3-ways TCP hand-shake successfully before the IDS would see this data packet. So the internal systems were actually running web servers.

Item 2:

1. Very likely these were Microsoft RPC exploit. A normal data transfer should not contain that many noop code.
2. Assuming this is TCP, the RPC was actually running on these systems. There were four internal servers involved. In general, this is not a good idea to expose this service on the Internet.

Item 3:

1. External 216.65.73.26 was exploiting nine internal seemingly ldap servers.
2. This external site belongs to flashhost, a adult web hosting service provider. There should not be any business relationship between the University and this provider.
3. Why ldap was involved could have many reasons such as this could be a VPN gateway use or the ldap server was unintentionally exposed to the Internet.

Item 4:

1. Port 445 is used by Microsoft system to send SMB(Server Message Block) messages over TCP.
2. These two internal IP showed up in port 135 use as well. This further proves that these systems are running Microsoft OS and are accessible from the Internet. No a good security thing.

Item 5:

1. I can speculate many possibilities on TCP port 1025 - Microsoft Remote Procedure Call (RPC) service, Blackjack network, trojan and etc.

Item 6:
1. Port 5000 is used for Windows Universal plug and play service (UPNP). There are exploit for the UPNP service.

Item 7:
1. Port 6129 is used by the Dameware remote administration software.
2. Older versions of Dameware can allow for unauthorized login and hence unauthorized use of Dameware for remote administration of a computer. Dameware was installed by some viruses for the purpose of remote administration of the infected system. (http://www.linklogger.com/TCP6129.htm)
3. Five system involved.

Item 8:
1. This contains a collection of randomly accessed port. Fully analyzing them will take too much time.

Concerns
1. Http traffic false positive is a concern.
2. Internet access to internal servers.
3. Unauthorized services running in internal servers

Recommendations
1. The IDS system could probably further fine tune to reduce the false positive on the port 80 traffic.
2. The University should review their policy on the access to the following systems from the Internet:
    - Windows OS systems.
    - ldap system.
3. The systems that runs Dameware should be examined

Alert #2 - MY.NET.30.3 activity
Alert rule
Unknown, probably custom rule.

Analysis
First of all, I ran the following command to summarize this alert type.

```
cat alert.all | grep 'MY.NET.30.3 activity' | sed 's/ \[\*\*\] /|/g' | cut -d "|" -f 3 | gawk
'{ split($1, T, ":"); print T[1], $2, $3;}' | sort | uniq -c | sort -nr
```

Sample Result:.
2166 131.92.177.18 -> MY.NET.30.3:524
1625 68.57.90.146 -> MY.NET.30.3:524
1558 69.138.77.62 -> MY.NET.30.3:524

The target ports were:
21, 80, 389, 427, 443, 446, 524, 554, 715, 1025, 1080, 1433, 2745, 2812, 3019, 3128, 3389, 3410, 4000, 4899, 5000, 6129, 8000, 9898, 12849, 20168, 55838

To get a better picture around My.NET.30.3, I ran the following command to check alerts other than MY.NET.30.3 activity.

cat alert.all | grep 'MY.NET.30.3' | grep -v 'MY.NET.30.3 activity'

Results:
04/08-08:13:54.361238 [**] NMAP TCP ping! [**] 216.54.131.241:80 -> MY.NET.30.3:427
04/08-08:13:54.394632 [**] NMAP TCP ping! [**] 216.56.88.61:81 -> MY.NET.30.3:427

I can't see exactly why the particular system was monitored so closely. A lot of services might be running on this system. Is this a honey pot of some sort.

Concerns
If there's rule specifically written for it, it must be monitored closely and I did not see other malicious traffic outside those being monitored. I therefore tend to believe there is not much we should be worrying about this system.

Recommendations
1. If it's a honey pot installed with a legitimate reason, just to make sure it's properly isolated from the normal production network and monitored closely. Once it's purpose is done, it should be removed from the network immediately.
2. If installed with no legitimate reason, remove it immediately.

Alert #3 - SMB Name Wildcard
Alert rule
alert udp any any -> $HOME_NET 137 (msg:"SMB Name Wildcard"; content:"CKAAAAAAAAAAAAAAAAAAAAAAAAAAAA|0000|";)

This is a very well known rule for detecting this activity.

Analysis
First of all, I ran the following command to summarize this alert type.

cat alert.all | grep 'SMB Name Wildcard' | sed 's/ \[**\] /|/g' | cut -d "|" -f 3 | gawk '{ split($1, T, ":"); print T[1], $2, $3;}' | sort | uniq -c | sort -nr

Sample result:
5185 MY.NET.11.7 -> 169.254.0.0:137

1831 MY.NET.11.7 -> 169.254.25.129:137
991 MY.NET.111.228 -> 209.2.144.10:137

According to http://archives.neohapsis.com/archives/snort/2000-01/0222.html, Windows machines typically send these types of queries in normal operation, particularly when file sharing is active, to determine NetBIOS names when only IP addresses are known. This type of query, when originating from an external network, is usually a pre-attack probe to gather netbios name table information such as workstation name, domain, and a list of currently logged in users.

The source IP were all from MY.NET. About 70% of the destination IP is inside 169.254.0.0/16. Running whois reveals the following information. It's IANA.

whois 169.254.0.0
[Querying whois.arin.net]
[whois.arin.net]

OrgName: Internet Assigned Numbers Authority
OrgID: IANA
Country: US
NetRange: 169.254.0.0 - 169.254.255.255
CIDR: 169.254.0.0/16
Comment: Please see RFC 3330 for additional information.

According to http://www.faqs.org/rfcs/rfc3330.html, 169.254.0.0 is used for the following purpose.

Quoted: '169.254.0.0/16 - This is the "link local" block. It is allocated for communication between hosts on a single link. Hosts obtain these addresses by auto-configuration, such as when a DHCP server may not be found.'

After discounting the DHCP IP 169.254.0.0/16, I still have to look at the following ones:

Source IP, MY.NET	**Dest IP**	**Count**
150.198	Internet, wide range	674
150.44	Internet, wide range	632
190.95	Internet, wide range	285
75.13	Internet, wide range	598

I do not know what's the real purpose of the traffic generated by these internal systems. They may be victim machines themselves or someone from inside were practicing hacking.

To check if the systems with these source IP have been involved with other malicious activities, the following command was run:

grep '<source_ip> alert.all | grep -v 'SMB Name Wildcard'. The “grep -v” filtered the alerts generated by the SMB rule.

MY.NET.150.198:
04/10-12:28:10.327710 [**] Null scan! [**] 216.250.41.184:33365 -> MY.NET.150.198:38598
04/10-12:28:10.327710 [**] Null scan! [**] 216.250.41.184:33365 -> MY.NET.150.198:38598

MY.NET.150.44 (sample output only):
04/11-10:10:49.061779 [**] EXPLOIT x86 NOOP [**] 203.218.215.28:3384 -> MY.NET.150.44:80
04/11-10:10:49.082775 [**] EXPLOIT x86 NOOP [**] 203.218.215.28:3384 -> MY.NET.150.44:80

MY.NET.190.95 (sample output only):
04/10-03:34:36.840094 [**] External RPC call [**] 217.160.94.163:111 -> MY.NET.190.95:111
04/10-04:09:10.919635 [**] EXPLOIT x86 NOOP [**] 61.230.239.29:4122 -> MY.NET.190.95:135
04/10-16:08:33.162323 [**] Possible trojan server activity [**] 213.189.89.109:2998 -> MY.NET.190.95:27374
04/11-02:57:47.951482 [**] SMB C access [**] 155.239.84.234:1136 -> MY.NET.190.95:139

MY.NET.75.13:
04/07-21:07:44.903489 [**] EXPLOIT x86 NOOP [**] 218.145.182.190:2596 -> MY.NET.75.13:80
04/09-06:18:02.620337 [**] NMAP TCP ping! [**] 194.75.139.2:80 -> MY.NET.1.3:53
04/10-00:45:31.475621 [**] MY.NET.30.3 activity [**] 210.75.133.105:4729 -> MY.NET.30.3:3128
04/11-07:15:12.028795 [**] EXPLOIT x86 NOOP [**] 68.43.170.140:3736 -> MY.NET.75.13:80

Concern
Three out of the four systems identified had been compromised, probably with root shell. This enables remote access through back door setup after the compromise.

Recommendations
1. Check and clean systems identified with compromise.

2. If the external hacker IP could be identified, block them at the gateway if possible. I know this may not be effective as the attacker can change IP pretty easily. But at least they should be blocked for some time.
3. Review the IDS system for the activities generated by these attacker to eradicate any other compromises they might have successfully done.

Alert #4 - High port 65535 tcp - possible Red Worm – traffic
Alert rule: I could not find this rule either from the current rule set or in the Internet, including snort website. I believe this rule simply detect traffic from port 65535.

Analysis
First of all, I ran the following command to summarize this alert type.

```
cat alert.all | grep 'Red Worm' | sed 's/ \[\*\*\] /|/g' | cut -d "|" -f 3 | gawk '{ split($1, T, ":"); print T[1], $2, $3;}' | sort | uniq -c | sort -nr
```

Sample Result:
```
 2168 MY.NET.60.16 -> 141.157.102.155:65535
618 MY.NET.97.51 -> 24.5.46.4:65535
608 62.77.191.33 -> MY.NET.153.83:1330
```

According to
http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html#p65535RW,
The "Red Worm" (not to be confused with "Code Red") is a Linux-specific infection, typified by a back door listening on port 65535 (which otherwise should not be used). Traffic seen from this port strongly indicates a compromised host. It's also aliased as Adore.

Other reference:
http://www.europe.f-secure.com/v-descs/adore.shtml
http://www.sans.org/y2k/adore.htm

According to the f-secure description, http://www.europe.f-secure.com/v-descs/adore.shtml:

Adore is a worm, that spreads in Linux systems using four different, known vulnerabilities already used by Ramen and Lion worms. These vulnerabilities concern BIND named, wu-ftpd, rpc.statd and lpd services.

I used spreadsheet to analyze the summary data to locate internal systems possibly be infected with the Red Worm as shown in the list below:
MY.NET.102.131
MY.NET.12.6
MY.NET.152.174

MY.NET.153.35
MY.NET.153.83
MY.NET.24.20
MY.NET.24.44
MY.NET.25.67
MY.NET.34.14
MY.NET.34.5
MY.NET.60.16
MY.NET.6.7
MY.NET.82.43
MY.NET.84.234
MY.NET.84.234
MY.NET.84.235
MY.NET.84.235
MY.NET.84.235
MY.NET.97.106
MY.NET.97.213
MY.NET.97.51
MY.NET.97.51
MY.NET.97.51
MY.NET.97.92
MY.NET.97.92

A total of 25 infected systems.

Concern
The Red Worm/Adore targets BIND named, wu-ftpd, rpc.statd and lpd services which are common services offered inside University campus. If this is not dealt with in a timely manner, the Red Worm could spread very quickly when unleashed.

Recommendations
1. Check, clean and patch up the identified 25 systems.
2. Block port 65535 at the border of the University.

Alert #5 - DDOS mstream handler to client
Alert rule
alert tcp $HOME_NET 12754 -> $EXTERNAL_NET any (msg:"DDOS mstream handler to client"; content: ">"; flow:to_client,established; reference:cve,CAN-2000-0138; classtype:attempted-dos; sid:248; rev:2;

Analysis
First of all, I ran the following command to summarize this alert type.

cat alert.all | grep 'DDOS mstream handler to client' | sed 's/ \[**\] /|/g' | cut -d "|"

-f 3 | gawk '{ split($1, T, ":"); print T[1], $2, $3;}' | sort | uniq -c | sort -nr

There are 7 items in the summary list:

3240 MY.NET.84.235 -> 82.48.242.184:4662
5 MY.NET.60.17 -> 65.54.252.99:25
3 MY.NET.84.235 -> 62.42.66.52:4662
2 MY.NET.84.235 -> 81.102.85.92:4662
1 MY.NET.84.235 -> 81.69.163.174:4662
1 MY.NET.84.235 -> 80.15.47.94:4662
1 MY.NET.84.235 -> 217.236.97.47:4662

According to http://www.giac.org/practical/Michael_Murphy_GCIH.doc, Mstream is a three-tiered DDoS tool, allowing an attacker to direct systems that have been infected with the mstream agent to flood target system(s) with sustained bursts of TCP packets, which significantly slows down the host by overburdening the CPU and even restricts network bandwidth.

These traffic did not use the same ports as shown in the CERT site (http://www.cert.org/incident_notes/IN-2000-05.html):
intruder --------- 6723/tcp -> handler
handler --------- 7983/udp -> agent
agent --------- 9325/udp -> handler

I was not able to find information pertaining the use of port 4662 in the communication between the handler and agent. This is very the port setting for a variant of the mstream exploit.

Other reference:
CIAC. “mstream DDOS tool” 3 May 2000
URL: http://www.ciac.org/ciac/bulletins/k-037.shtml

Concerns
A massive DDOS against the University may have potential to grind their network to a complete halt. The University should take a very serious stand on this to clean up the compromised systems.

Recommendations
1. Check the following systems for potential mstream infection:
   MY.NET.84.235
   MY.NET.60.17
2. Block the following external IP:
   82.48.242.184
   65.54.252.99
   81.102.85.92

217.236.97.47

Alert #6 - Possible trojan server activity
Alert rule: I could not find this rule either from the current rule set or in the Internet, including snort website. I believe this rule simply detect port 27374 activities.

Analysis
First of all, I ran the following command to summarize this alert type.

cat alert.all | grep 'Possible trojan server activity' | sed 's/ \[**\] /|/g' | cut -d "|" -f 3 | sort | uniq -c | sort -nr

This command line is somewhat different from the other as I want to see the source as well. To do this, I removed the gawk portion.

Sample result:
38 68.55.195.232:27374 -> MY.NET.12.6:25
31 MY.NET.12.6:25 -> 68.55.195.232:27374
16 MY.NET.24.44:80 -> 170.91.5.4:27374

According to Glenn,
http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html#posstroj is exclusively triggered on port 27374 which is commonly used by trojans.

This alert was studied in Hee's paper, http://www.sans.org/rr/papers/23/836.pdf

Concerns
As port 27374 is commonly used by trojan. This means that system runs with this port may indicate the existence of trojan on system.

Feeding the summary data generated above into a spreadsheet, I found the following interesting points:

1. I believe there were some false positives. There are normal 27374 client port used for network traffic accessing external email, web, POP and SSL servers.

Example:

MY.NET.12.6	SMTP
MY.NET.24.34/44/11	HTTP
MY.NET.12.4	POP3
MY.NET.24.74	SSL

2. External IP 213.189.89.54 might have control of a large number of systems in the MY.NET.190.0/24 subnet. It was using sequentially incremented source port

to access port 27374 on the machine in this internal subnet.
Example:
213.189.89.54 port 1181 -> 4922 -> MY.NET.190.1 -> 254
213.189.89.109 port 1862 -> 4458-> MY.NET.109.1 -> 254

Both these systems run on the qualitynet.net network.

Recommendations
1. Check the internal systems on internal subnet My.NET.190.0/24 listed above for Possible trojan server activity.
2. Block the following external IP listed above.
   1. 213.189.89.54
   2. 213.189.89.109
3. Fine tune the IDS to filter normal activities using port 27374 to reduce false positive.

Alert #7 - External RPC call
Alert Rule – I could not find this rule either from the current rule set or in the Internet, including snort website. I believe this rule simply detect port 111 activities.

Analysis
First of all, I ran the following command to summarize this alert type.
cat alert.all | grep 'External RPC call' | sed 's/ \[**\] /|/g' | cut -d "|" -f 3 | gawk '{ split($1, T, ":"); print T[1], $2, $3;}' | sort | uniq -c | sort -nr

Sample result:
```
      8 213.46.246.46 -> MY.NET.6.15:111
    3 213.46.246.46 -> MY.NET.5.5:111
    3 213.46.246.46 -> MY.NET.190.99:111
```

Further analyzing the table generated above using spreadsheet, I found that there are only two major external attackers: 213.46.246.46 and 217.169.94.163. The internal subnet that they probed or accessed is MY.NET.190.0/16.

nslookup reveals:
213.46.246.46 - magyar.chello.hu
217.169.94.163 – failed to resolve name

To check for other clues on the attacks, I ran more check around these IP in the alert, oos and scan log. The basic command is grep '<ip>' <log type> | grep -v 'External RPC call'. The example for alert log is grep '213\.46\.246\.46' alert.all | grep -v 'External RPC call'.

Alert log result:
213.46.246.46 – port scan
217.169.94.163 – nothing found

OOS log result:
213.46.246.46 – nothing found
217.169.94.163 – nothing found

Scan log result:
213.46.246.46 – sync scan port 111 only
217.169.94.163 – sync scan port 111 only

The same alert was discussed in the following papers:
Menke, Mark. "GCIA Practical Assignment V2.2.5"
URL: http://www.giac.org/practical/Mark_Menke_GCIA.doc

Concerns
After reviewing all the logs, I don't think there's immediate threat to the University from this activity. These two external attackers had only launched scans on the RPC port. It doesn't seem to me anything had been compromised as a result. If any of these target systems was compromised, I should see some other connections to the high ports on the target systems from the attacker systems. But this was not the case. So I could safely say that the University was safe this round.

Recommendations
Safe this time does not necessary be good forever. So the University should take the following precautions to make sure:
1. Keep the system patches up to date for those system that offer RPC services.
2. Review and only allow necessary connection to the outside world.
3. Block the two external attacker IP for some period of time. It's very easy to change their IP.

Alert #8 - FTP passwd attempt
Alert rule: I could not find this rule either from the current rule set or in the Internet, including snort website.

Analysis
First of all, I ran the following command to summarize this alert type.

```
cat alert.all | grep 'FTP passwd attempt' | sed 's/ \[\*\*\] /|/g' | cut -d "|" -f 3 | gawk
'{ split($1, T, ":"); print T[1], $2, $3;}' | sort | uniq -c | sort -nr
```

Sample result:

3 202.20.73.30 -> MY.NET.24.47:21
3 198.204.133.209 -> MY.NET.24.47:21
2 69.34.61.222 -> MY.NET.24.47:21

Further analyzing the table generated above using spreadsheet, I found that this seems to be a simple case as only one internal systems involved, MY.NET.24.47. The attacking systems were random from the Internet.

To check for other clues on this attacks, I ran more check around these IP in the alert, oos and scan log. The basic command is grep '<ip>' <log type> | grep -v 'FTP passwd attempt'. The example for alert log is grep '\.24\.47' alert.all | grep -v 'FTP passwd attempt'.

Alert log result:
MY.NET.24.47 – port scanned others and being null scanned by 4.8.204.245.

OOS log result:
MY.NET.24.47 – always be the target.

Scan log result:
MY.NET.24.47 – being scanned by external and internal system 130.85.150.199. This internal system seems to be infected with the Agobot/Gaobot worm. See scan analysis section below for information about this worm.

Apr 7 15:36:41 130.85.150.199:1700 -> 130.158.24.47:2745 SYN ******S*
Apr 7 15:36:41 130.85.150.199:1701 -> 130.158.24.47:135 SYN ******S*
Apr 7 15:36:41 130.85.150.199:1702 -> 130.158.24.47:1025 SYN ******S*
Apr 7 15:36:41 130.85.150.199:1703 -> 130.158.24.47:445 SYN ******S*
Apr 7 15:36:41 130.85.150.199:1704 -> 130.158.24.47:3127 SYN ******S*
Apr 7 15:36:41 130.85.150.199:1705 -> 130.158.24.47:6129 SYN ******S*
Apr 7 15:36:41 130.85.150.199:1706 -> 130.158.24.47:139 SYN ******S*
Apr 7 15:36:41 130.85.150.199:1707 -> 130.158.24.47:80 SYN ******S*

Concerns
I don't believe there's immediate concern from this alert.

Recommendations
Safe this time does not necessary be good for ever. So the University should take the following precautions to make sure:
1. Keep the patches up to date for those system that offer the ftp services.
2. Review and only allow necessary connection to the outside world.

Alert #9 - IRC evil - running XDCC
Alert rule: I could not find this rule either from the current rule set or in the Internet, including snort website. It's very likely that this is a custom rule.

Analysis
First of all, I ran the following command to summarize this alert type.

cat alert.all | grep 'IRC evil - running XDCC' | sed 's/ \[**\] /|/g' | cut -d "|" -f 3 | gawk '{ split($1, T, ":"); print T[1], $2, $3;}' | sort | uniq -c | sort -nr

There only 4 items in the summary list:
```
     60 MY.NET.43.2 -> 64.246.60.72:6667
     10 MY.NET.82.79 -> 207.36.180.241:6663
      1 MY.NET.82.79 -> 207.36.180.241:6669
      1 MY.NET.43.7 -> 64.62.196.26:6667
```

nslookup for the target systems:
64.62.196.26 - purplecone.widge.net
64.246.60.72 - dddbox.triple-d.us
207.36.180.241 – no name resolved.

According to Donald's paper (http://www.giac.org/practical/GCIA/Donald_Merchant_GCIA.doc), the alert could indicate the following:
1. Computers running XDCC may be remotely controlled by people in the IRC channels.
2. The hacked machines were found to be running a back door remote access service, also called XDCC.
3. This additional back door allows the intruder to access the machine as an administrator from a remote computer.

Al's paper also analyzed this alert (http://www.whitehats.ca/main/members/Herc_Man/Files/Al_Williams_GCIAPractical.pdf).

Concern
ICQ in general is having lot of security problems. It's home to lot of blackhats.

Recommendations
There is not much choices on security solution that are effective, less costly and yet not prohibiting the free information exchange environment in an University.

I have given thoughts to the following mitigation but gave up:
1. Restrict access to ICQ at the gateway. This is too disturbing.
2. Security awareness promotion on the use of ICQ and other instant messaging

solutions. I really doubt its effectiveness. Be realistic, how many students will be coming to listen.

I believe the best protection that the University should do is to segregate the teaching stuff and student network from the networks that host the critical infrastructure of their IT systems.

Alert #10 - RFB - Possible WinVNC – 010708-1
Alert rule: I could not find this rule either from the current rule set or in the Internet, including snort website. It's very likely that this is a custom rule.

Analysis
First of all, I ran the following command to summarize this alert type.

```
cat alert.all | grep 'RFB - Possible WinVNC' | sed 's/ \[\*\*\] /|/g' | cut -d "|" -f 3 | sort | uniq -c | sort -nr
```

This command line is somewhat different from the other as I want to see the source as well. To do this, I removed the gawk portion.

Sample result:

```
 1 MY.NET.84.231:5901 -> 141.157.72.181:3873
1 MY.NET.84.231:5901 -> 141.157.72.181:3157
1 MY.NET.70.225:5900 -> 68.55.111.191:3606
```

According to description in http://www.csd.uwo.ca/staff/magi/doc/vnc/start.html, the default port of VNC server is at 5900 for display number zero, 5901 for display number one. VNC client can choose different display number to connect.

Further analyzing the table generated above using spreadsheet, I found one external VNC server but several internal ones.

```
1 MY.NET.53.31 3791 -> 68.55.111.196 5900
1 MY.NET.53.44 3047 -> 68.55.111.196 5900
1 68.55.205.88 4138 -> MY.NET.111.46 5900
1 68.55.205.88 4379 -> MY.NET.111.46 5900
1 68.55.192.251 60253 -> MY.NET.111.51 5900
1 68.55.192.251 63445 -> MY.NET.111.51 5900
1 68.55.192.251 62902 -> MY.NET.111.51 5900
1 68.55.192.251 62931 -> MY.NET.111.51 5900
1 68.55.111.191 3606 -> MY.NET.70.225 5900
1 68.55.111.191 3240 -> MY.NET.70.225 5900
1 141.157.72.181 3873 -> MY.NET.84.231 5901
1 141.157.72.181 3157 -> MY.NET.84.231 5901
1 141.157.72.181 3362 -> MY.NET.84.231 5901
```

Concerns
VNC could both be a legit server or a trojan. The major concerns are:
1. The VNC server is actually trojan or illegal system back door.
2. If it's legitimate service, how secure it is.

Recommendations
1. Check the following systems if the VNC servers are legitimate.
MY.NET.111.46
MY.NET.111.51
MY.NET.70.225
MY.NET.84.231
2. Restrict access to internal VNC service.

OOS Logs Analysis

OOS logs capture packet with non-standard TCP flag bits. This could be the results of new protocol features, network problem and malicious activities. Malicious activities can range from system reconnaissance, finger-printing, system invasion and etc.

I wrote a little script called scanoos [24] to summarize the five days OOS log down to the source, destination and the flag bit combination.

From the defense point of view, I sorted destination IP of the OOS summary. This gave me the view of looking out from the University. I picked out the summary traffic that I have concerned and listed them immediately below.

Cnt	Source	Destination	Flag
1	207.217.120.232	MY.NET.12.6:25	12**PRS*
1	211.253.127.81	MY.NET.12.6:25	12UAPR**
1	211.74.6.132(211-74-6-132.adsl.dynamic.seed.net.tw.)	MY.NET.12.6:25	******SF
1	4.5.82.74	MY.NET.12.6:25	*2U***SF
1	64.110.199.1	MY.NET.12.6:25	12UAPRS*
1	66.218.66.79	MY.NET.12.6:25	********
2	200.59.101.29(modem29-as1.comodoro.sinectis.com.ar.)	MY.NET.153.35:3247	********
1	68.167.207.243(h-68-167-207-243.snfccasy.dynamic.covad.net.)	MY.NET.153.35:3247	***AP*SF
1	68.167.207.243(h-68-167-207-243.snfccasy.dynamic.covad.net.)	MY.NET.153.35:3247	**U*****
1	68.167.207.243(h-68-167-207-243.snfccasy.dynamic.covad.net.)	MY.NET.153.35:3247	**U*P*SF
1	68.167.207.243(h-68-167-207-243.snfccasy.dynamic.covad.net.)	MY.NET.153.35:3247	*2*A*RSF
1	68.167.207.243(h-68-167-207-243.snfccasy.dynamic.covad.net.)	MY.NET.153.35:3247	*2U***SF
1	68.167.207.243(h-68-167-207-243.snfccasy.dynamic.covad.net.)	MY.NET.153.35:3247	*2UA*RSF
1	68.7.138.89(ip68-7-138-89.sd.sd.cox.net.)	MY.NET.153.35:3247	12U*PR**
1	82.72.190.47(cm53719-a.maast1.lb.home.nl.)	MY.NET.153.35:3247	**U*P*SF
1	138.23.236.133(ultrafast6.ucr.edu.)	MY.NET.24.47:1063	12*****F
1	138.23.236.133(ultrafast6.ucr.edu.)	MY.NET.24.47:1063	12*A*R**
1	138.23.236.133(ultrafast6.ucr.edu.)	MY.NET.24.47:1068	******SF
1	138.23.236.133(ultrafast6.ucr.edu.)	MY.NET.24.47:1076	********
1	138.23.236.133(ultrafast6.ucr.edu.)	MY.NET.24.47:1079	*2U***SF
3	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3964	********
2	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3964	**U*****
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3964	1***P*SF
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3964	12*A**SF
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3964	12*AP*SF
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3964	12U***SF
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3964	12U*PR*F
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3964	12U*PRS*
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3964	12UAP*S*
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3964	12UAPR*F
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3964	12UAPRSF
9	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	********
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	****PRSF
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	**U**RSF
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	1**AP*SF
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	1*U*PRSF
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	1*UA*RSF
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	12****SF
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	12***R**
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	12**P*S*
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	12**PRS*
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	12*AP***
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	12*APRS*
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	12U**RS*
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	12U*P***
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	12U*PRSF
1	4.8.204.245(wbar1.lax1-4-8-204-245.dsl-verizon.net.)	MY.NET.24.47:3970	12UA*RSF

1. Most of the flag settings in the table above have bits 1 (CWR, Congestion

Window Reduced) and bit 2 (ECN-E, Explicit Congestion Notification Echo) turned on together with other flag bit.
2. The flags with bit 1, 2 and S (syn) turned on have been filtered off already because this combination could be a legitimate use.
3. CWR and ECN-E and SYN will be turned on at the same time to negotiate ECN operation. However, some OS fingerprinting scanner such as QUESO also use this combination to scan system. The way to tell the difference is to look at the whole state to see if the full ECN exchange was actually completed or just only seeing this combination.

The valid combinations according to the GCIA training material are:

Condition	TCP-CWR	TCP-ECE
SYN	1	1
SYN-ACK	0	1
ACK	0	0
No Congestion	0	0
Congestion	0	0
Receiver Response	0	1
Sender Response	1	1

Most of flag combination in the above table are not valid. This makes me to believe that either they were used to fingerprint systems or to evade firewall or IDS systems.

Realistically, system reconnaissance and fingerprinting are very normal activities these days on the Internet. The important consideration is not how to stop them but how to best protect yourself by not leaking system information to the wrong hands.

Concern
Someone will eventually find a way to compromise these systems which could be used as a jumping board to hurt the University, to hurt others or use them to host inappropriate materials/services on the Internet.

Is it possible that some of these systems had been compromised already? I checked the destination addresses against the alert and scan logs by running “grep '\.153\.35' alert.all “.

The alert log review result may suggest that MY.NET.153.35 was compromised and became an evil machine:
04/07-23:07:35.457071 [**] EXPLOIT x86 NOOP [**] 63.199.127.11:17145 -> MY.NET.153.35:1025
04/08-10:23:31.250345 [**] EXPLOIT x86 NOOP [**] 67.101.252.13:4250 -> MY.NET.153.35:1025

04/08-10:41:46.150726 [**] EXPLOIT x86 NOOP [**] 68.248.29.158:1104 -> MY.NET.153.35:80
04/08-12:11:38.684370 [**] EXPLOIT x86 setgid 0 [**] 81.76.69.63:2109 -> MY.NET.153.35:3247
04/08-16:30:30.467324 [**] EXPLOIT x86 setuid 0 [**] 65.28.248.111:2679 -> MY.NET.153.35:3247
04/08-17:06:32.350115 [**] EXPLOIT x86 setuid 0 [**] 24.162.203.193:4410 -> MY.NET.153.35:3247
04/08-18:34:06.225704 [**] spp_portscan: PORTSCAN DETECTED from MY.NET.153.35 (THRESHOLD 12 connections exceeded in 6 seconds) [**]
04/08-18:34:09.811632 [**] spp_portscan: portscan status from MY.NET.153.35: 18 connections across 18 hosts: TCP(3), UDP(15) [**]

As shown in the sample log above, MY.NET.153.35 were exploited in the following sequence and then it started to scan the network crazily.
1. EXPLOIT x86 NOOP
2. EXPLOIT x86 setuid 0
3. Then it generated lots of port scanning activities.

Looks like port 3247 was an open shell port setup after the system was compromised. Google search did not return any information that indicates this is a port used in common exploit. I did a search on Dshield port information (http://ww.dshield.org/port_report.php), the volume of its use around the period of these logs were very low at about 50 records a day as compared to a daily of 100-150 K of Mydoom port 3127 records. This indicates that the port has not been widely used so far.

External system 68.167.207.243 has scanned the University systems but no other exploit was alerted. I suspected that it was just able to locate port 3247 left behind by the previous attackers.

Recommendations
1. Verify if these services are really needed to be open to the Internet.
2. If yes, make sure they are running with sufficient network protection, up-to-date system patches, idle user accounts are removed and best security practices are followed to secure the systems.
3. If not needed, they should be shut down immediately.

Correlation
1. Fyodor. “Remote OS detection via TCP/IP Stack FingerPrinting” 18 Oct 1998 URL: http://www.insecure.org/nmap/nmap-fingerprinting-article.txt
2. So, Hee. “GCIA Practical Assignment v 3.0”. 16 Feb 2002. URL: http://www.sans.org/rr/papers/23/836.pdf
3. ZVON. “The Specific Example of ECN”

URL: http://www.zvon.org/tmRFC/RFC3360/Output/chapter3.html
4. GCIA training material, Network Traffic Analysis Using Tcpdump, page 5-21.
5. Warner, Neil. “Scan 23”
URL: http://project.honeynet.org/scans/scan23/sol/Neil.html. Quoted 'At packet #150759 a TCP XMAS Tree scan starts. A XMAS tree scan sets the FIN, URG and PSH flags. If a port is closed and receives the XMAS tree scan the target will send back a RST. '

Scan Log Analysis

I ran the following command to get a feel how much scanning activities there and get a summary out of all the scan logs.

cat scans.all | awk -F" " '{ split($4, TEMP, ":"); split($6, TEMP1, ":"); print TEMP [1],$5,TEMP1[2],$7; }' | sort | uniq -c | sort -nr >> scans.sum.nosrcport.nodstip

But there's no MY.NET.x.x IP addresses in the scan log. Googled 'MY.NET' & '130.83' returned the following link http://www.dshield.org/pipermail/intrusions/2003-February/006925.php which confirmed my thinking that '130.83' equals to 'MY.NET'.

There are 1308 items found. It's not realistic to list all. I import the space separated file to spreadsheet to sort the source IP to look for scan pattern. The patters were summarized in the table immediately below.

Count	Src IP, MY.NET.	Src Port	Dst Port	Proto	Remark
2878693	1.3	32783	53	UDP	DNS server, critical infrastructure
786948	1.4	32788	53	UDP	DNS server, critical infrastructure
656515	111.34; 153.35	-	high ports, 1024 – 65535, not all	UDP	UDP port scan

Count	Src IP, MY.NET.	Src Port	Dst Port	Proto	Remark
3933721	66.56; 70.96; 80.224; 80.5; 84.145; 84.224; 12.152; 150.199; 150.210; 151.75; 42.2; 43.10; 43.5; 153.174	-	135,139,445,1025,2745,3127,3410,5000,6129	TCP	Agobot/Gaobot worm.

DNS Server Scanning

The default source UDP port for DNS queries is also 53. But MY.NET.1.3 and MY.NET.1.4 used 32783 and 32788. The DNS source port setting can be set to any port greater than 1023. See excerpt from http://www.oreillynet.com/pub/a/network/excerpt/dnsbindcook_ch07/ below:

(On BIND 8 name servers, *query-source* also controls the source port for NOTIFY messages and refresh queries.) For example, to instruct a name server to use port 1053 as the source port for all outbound queries, use:

```
options {
    directory "/var/named";
    query-source address * port 1053;
};
```

There's another possibility that the traffic was generated by malicious code as the source port was fixed. I checked into the alert log to see if there's any sign of system compromise. I ran the following command:

```
grep 'MY\.NET\.1\.3' alert.all | grep -v 'portscan' | grep -v 'NMAP TCP ping' |  cut -d " " -f 2- | sort | uniq -c | sort -nr
```

Results:

```
11 [**] EXPLOIT x86 NOOP [**] 61.207.234.254:3278 -> MY.NET.1.3:80
10 [**] EXPLOIT x86 NOOP [**] 66.32.128.69:3505 -> MY.NET.1.3:80
8 [**] EXPLOIT x86 NOOP [**] 64.219.108.66:4673 -> MY.NET.1.3:80
7 [**] EXPLOIT x86 NOOP [**] 61.109.238.42:4858 -> MY.NET.1.3:80
5 [**] EXPLOIT x86 NOOP [**] 210.106.239.115:2114 -> MY.NET.1.3:80
4 [**] EXPLOIT x86 NOOP [**] 24.235.213.166:1274 -> MY.NET.1.3:80
1 [**] EXPLOIT x86 NOOP [**] 203.70.116.4:4347 -> MY.NET.1.3:80
11 [**] EXPLOIT x86 NOOP [**] 61.207.234.254:3278 -> MY.NET.1.3:80
```

So this DNS server has been attacked that there's chance the fix source port could be crafted.

But I lean toward to think that it's the University administrator picked this port. However, the University should be prudent to check the systems to make sure there's no compromise.

Concern
DNS system MY.NET.1.3 and MY.NET.1.4 may have been compromised.

Recommendation
These two system should be investigated to make sure they have not be compromised.

UDP port scan
This seems to be scans from a compromised systems MY.NET.111.34 and MY.NET.153.35. The University should check these systems out.

To check if there's any correlation information exists in the alert log, run:
grep 'MY\.NET\.111\.34' alert.all | grep -v 'portscan' | cut -d " " -f 2- | sort | uniq -c | sort -nr

Sample results:
EXPLOIT x86 NOOP [**] 200.104.40.64:2658 -> MY.NET.111.34:80
EXPLOIT x86 setuid 0 [**] 61.49.164.14:4662 -> MY.NET.111.34:1585
EXPLOIT x86 setuid 0 [**] 80.38.200.21:1282 -> MY.NET.111.34:4662
TFTP - Internal UDP connection to external tftp server [**] MY.NET.111.34:4672 -> 80.53.47.2:69
NMAP TCP ping! [**] 65.71.55.244:80 -> MY.NET.111.34:4672
High port 65535 udp - possible Red Worm - traffic [**] MY.NET.111.34:4672 -> 207.248.43.224:65535

The sample results have some interesting observations:
1. Even this sample result did not have the same external IP, we can still try to put the puzzle together.
2. This system seems to be running a web server.
3. It was compromised by the “EXPLOIT x86 NOOP” buffer overflow exploit.
4. Once a shell was setup, the attacker gain root access by setting the uid to 0 which is root.
5. Then the attacker used tftp to download the other exploit code to the system. Then MY.NET.111.34 always used port 4672 as the source ports after the “EXPLOT” and “TFTP” alerts.

Concern
The identified systems may have a good chance that they have been

compromised.

Recommendations
These systems should be investigated. They should be cleaned if showing sign of compromise.

Agobot/Gaobot worm

The following systems were infected with Agobot/Gaobot or other variants:
MY.NET.66.56
MY.NET.70.96
MY.NET.80.224
MY.NET.80.5
MY.NET.84.145
MY.NET.84.224
MY.NET.12.152
MY.NET.150.199
MY.NET.150.210
MY.NET.151.75
MY.NET.42.2
MY.NET.43.10
MY.NET.43.5
MY.NET.153.174

Correlation
As quoted from FSecure webpage " The back door can scan subnets for exploitable computers and send a list of their IPs to the bot operator. The scan is performed on ports 80, 135 and 445 for RPC/DCOM (MS03-026), RPC/Locator (MS03-001) and WebDAV (MS03-007) vulnerabilities. The backdoor can also scan for computers infected with MyDoom worm (port 3127), Bagle worm (port 2745) and also for computers where DameWare remote system management software is installed (port 6129)."

F-Secure, "Agobot.FO" Mar 2004
URL: http://www.f-secure.com/v-descs/agobot_fo.shtml#details

Concern
This type of worm has very damaging capabilities. Leaving them uncleaned risks a future computer security incident outbreak.

Recommendation
These systems are very likely infected with the Agobot/Gaobot worm. They should be virus scanned and the system administrators should take appropriate action to clean the worm in case infected.

Top 10 talker

Alert logs

By Source IP and Volume
I ran the following command to get the top 10 IP:
cat alert.all | grep -v 'portscan' | sed 's/ \[**\] /|/g' | sed 's/ -> /|/g' | gawk -F "|"
'{ split($3, T, ":"); split($4, U, ":"); printf("%s\n", T[1]); }' | sort | uniq -c | sort -nr

Result (Count IP):
7559 212.76.225.24
7016 MY.NET.11.7
3954 MY.NET.84.235
3480 199.131.21.34
2993 68.81.0.87
2693 141.157.102.155
2169 MY.NET.60.16
2166 131.92.177.18
1660 68.57.90.146
1628 69.138.77.62

By Source IP, Alert Message and Volume
Then I included the alert message as well to see these top talkers were detected by more than one signature:
cat alert.all | grep -v 'portscan' | sed 's/ \[**\] /|/g' | sed 's/ -> /|/g' | gawk -F "|"
'{ split($3, T, ":"); split($4, U, ":"); printf("%s|%s\n", T[1], $2); }' | sort | uniq -c | sort -nr
Note: $2 added in the printf statement produce the alert message as well.

Result (Count Source IP|Alert Message):
7502 212.76.225.24|Tiny Fragments - Possible Hostile Activity
7016 MY.NET.11.7|SMB Name Wildcard
3479 199.131.21.34|EXPLOIT x86 NOOP
3250 MY.NET.84.235|DDOS mstream handler to client
2993 68.81.0.87|MY.NET.30.4 activity
2693 141.157.102.155|High port 65535 tcp - possible Red Worm - traffic
2168 MY.NET.60.16|High port 65535 tcp - possible Red Worm - traffic
2166 131.92.177.18|MY.NET.30.3 activity
1625 68.57.90.146|MY.NET.30.3 activity
1558 69.138.77.62|MY.NET.30.3 activity

By Source IP, Alert Message, Destination Port and Volume
Then I included also the destination port:
cat alert.all | grep -v 'portscan' | sed 's/ \[**\] /|/g' | sed 's/ -> /|/g' | gawk -F "|"
'{ split($3, T, ":"); split($4, U, ":"); printf("%s|%s|%s\n", T[1], $2, U[2]); }' | sort |

uniq -c | sort -nr
Note: U[2] is the destination port.

Result (Count Source IP|Alert Message|Destination Port):
  7502 212.76.225.24|Tiny Fragments - Possible Hostile Activity|
   7016 MY.NET.11.7|SMB Name Wildcard|137
   3248 MY.NET.84.235|DDOS mstream handler to client|4662
   2975 68.81.0.87|MY.NET.30.4 activity|51443
   2885 199.131.21.34|EXPLOIT x86 NOOP|80
   2693 141.157.102.155|High port 65535 tcp - possible Red Worm - traffic|22
   2168 MY.NET.60.16|High port 65535 tcp - possible Red Worm - traffic|65535
   2166 131.92.177.18|MY.NET.30.3 activity|524
   1625 68.57.90.146|MY.NET.30.3 activity|524
   1558 69.138.77.62|MY.NET.30.3 activity|524

It is interesting to see that the top 10 talkers remain the same for these three level of search. This tells me that they only generate the same traffic against the same target port. This may mean either they were infected with something or running some kind automatic systems such as network scan system.

OOS logs Analysis

First of all, I ran script scanoos [24] to normal the OOS log for easy handling. Its original format makes analysis using machine more difficult. The name of the normalized OOS log file I will be using is called oos.normal.

By Source IP and Volume
Command used: cat oos.normal | gawk '{ split($2,T,":"); print T[1];}' | sort | uniq -c | sort -nr

Result (Count IP):
   1707 68.54.84.49
    700 202.144.28.167
    259 141.224.64.4
    171 62.174.236.17
    169 193.170.194.27
    166 66.225.198.20
    163 80.54.249.136
    116 80.38.206.68
     96 MY.NET.199.202
     90 202.54.60.162

By Source IP, Alert Message and Volume
Command used: cat oos.normal | gawk '{ split($2,T,":"); printf("%s|%s\n", T[1], $12);}' | sort | uniq -c | sort -nr

Result (Count Source IP|Alert Message):
```
   1707 68.54.84.49|12****S*
  700 202.144.28.167|12****S*
  259 141.224.64.4|12****S*
  171 62.174.236.17|12****S*
  169 193.170.194.27|12****S*
  166 66.225.198.20|12****S*
  163 80.54.249.136|12****S*
  116 80.38.206.68|12****S*
   96 MY.NET.199.202|12****S*
   90 202.54.60.162|12****S*
```

By Source IP, Alert Message, Destination Port and Volume

Command: cat oos.normal | gawk '{ split($2,T,":"); split($4,U,":"); printf("%s|%s|%s\n", T[1], $12, U[2]);}' | sort | uniq -c | sort -nr

Result (Count Source IP|Alert Message|Destination Port):
```
  1707 68.54.84.49|12****S*|110
   700 202.144.28.167|12****S*|4662
   259 141.224.64.4|12****S*|25
   171 62.174.236.17|12****S*|24842
   168 193.170.194.27|12****S*|113
   166 66.225.198.20|12****S*|25
   163 80.54.249.136|12****S*|4662
   116 80.38.206.68|12****S*|28053
    90 202.54.60.162|12****S*|80
    86 68.121.194.43|********|110
```

The result is very similar to alert log except MY.NET.199.202 drops off the last top 10 list. So I ran the command again but only captured the output for this IP.

Result:
```
    41 MY.NET.199.202|12****S*|443
    35 MY.NET.199.202|12****S*|80
    10 MY.NET.199.202|12****S*|143
     8 MY.NET.199.202|12****S*|22
     2 MY.NET.199.202|12****S*|25
```

It falls within my expectation that this machine target more than one single destination ports.

Scan logs Analysis

By Source IP and Volume

Command used: cat scans.all | gawk '{ split($4,T,":"); print T[1]; }' | sort | uniq -c | sort -nr

Result (Count IP):
2890410 130.85.1.3
1623397 130.85.111.51
1522547 130.85.153.35
1189494 130.85.81.39
1130696 130.85.70.96
1082054 130.85.112.152
 796676 130.85.1.4
 338578 130.85.66.56
 295221 130.85.84.235
 253164 130.85.42.2

By Source IP, Scan Type and Volume
Command used: cat scans.all | gawk '{ split($4,T,":"); printf("%s|%s\n", T[1],$7); }' | sort | uniq -c | sort -nr

Result (Count Source IP|Scan Type):
2890110 130.85.1.3|UDP
1623293 130.85.111.51|SYN
1512393 130.85.153.35|UDP
1189337 130.85.81.39|SYN
1130693 130.85.70.96|SYN
1082027 130.85.112.152|SYN
 796630 130.85.1.4|UDP
 338570 130.85.66.56|SYN
 278697 130.85.84.235|SYN
 253157 130.85.42.2|SYN

By Source IP, Scan Type, Destination Port and Volume
Command used: cat scans.all | gawk '{ split($4,T,":"); split($6,U,":"); printf("%s|%s|%s\n", T[1],$7,U[2]); }' | sort | uniq -c | sort -nr

Result (Count Source IP|Scan Type |Destination Port):
2878693 130.85.1.3|UDP|53
1622974 130.85.111.51|SYN|135
1188912 130.85.81.39|SYN|135
 786948 130.85.1.4|UDP|53
 230195 130.85.34.14|SYN|25
 216911 130.85.84.235|SYN|4662
 214518 130.85.97.28|SYN|80
 203647 130.85.97.55|UDP|22321
 166819 130.85.112.152|SYN|2745

151458 130.85.112.152|SYN|135

Some systems scanned more than one port. The top talkers are the same for the first two reports. But four of them dropped off from the last report which also included the destination port. The only reason I can think of is that they scan more than one single point that their original count got thinned out.

I counted the number of different ports these systems had scanned:
130.85.153.35 : 7118
130.85.70.96 : 14
130.85.66.56 : 20
130.85.42.2 : 18

Reviewing the alerts logs for 130.85.153.35, it had been attacked by buffer overflow, Red Worm and different scans. I believe that this machine has been compromised very seriously.

The others were running IRC and the logs did not reveal any sign of compromise.

Five Selected External Source Addresses

These IP were picked from the list of external system identified with suspicious activities.

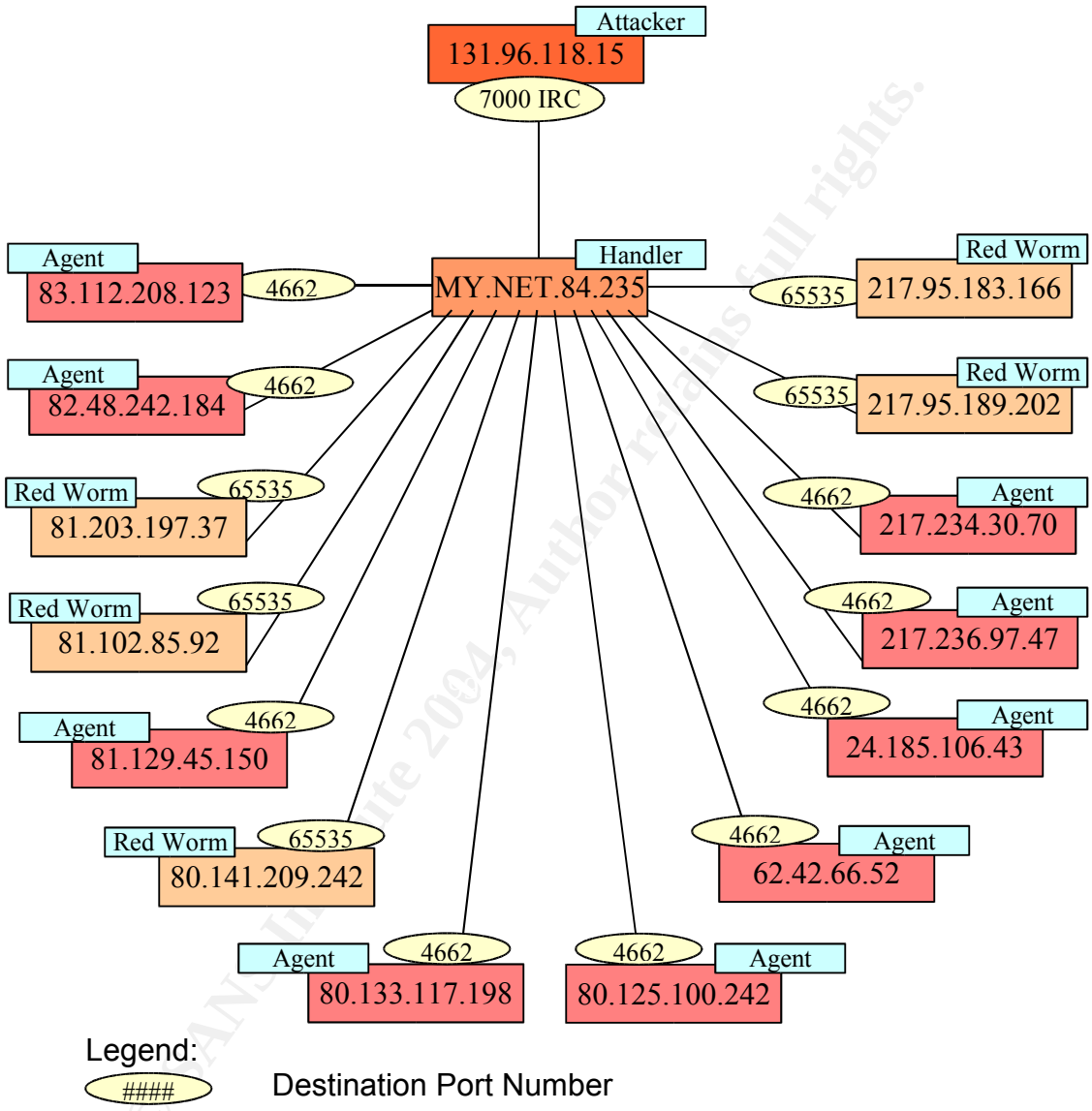
Information gathered from samspade.org and www.arin.net:

Host	Information	Contact
216.65.73.26	Host name: flashca249.flashhost.com Org Name: Maxim Country: US	TechName:Maxim Computer System Add: 42712 Lawrence Place, Fremont, CA, 94538. Ph:1-510-226-0695 Email: noc@maxim.net
131.92.177.18	Host name: aeclt-cf00a4.apgea.army.mil Org Name: Army Information System Command – Aberdeen(EA) Country: US	TechName: Ward Ronnie Add:AMSSB-SCI-N/BLDG E5234, Aberdeen Proving Ground, MD Ph: 1-410-436-4755 Email: ronnie.ward@sbccom.apgea.army.mil
141.157.102.155	Host name: pool-141-157-102-155.balt.east.verizon.net Org Name: Verizon Internet Services Country: US	TechName: Add: 1880 Campus Commons Dr, Reston, VA, 20191 Ph: 1-703-295-4583 Email: noc@gnilink.net
68.55.195.232	Host name: pcp230189pcs.catonv01.md.comcast.net Org Name: Comcast Cable Communications Inc. Country: US	TechName: Comcast Cable Communications Inc Add: 1800 Bishops Gate Blvd, Mt Laurel, NJ, 08054 Ph: 1-856-317-7200 Email: cips_ip-registration@cable.comcast.com
213.46.246.46	Host name: magyar.chello.hu Org Name: Chello Broadband Country: Netherlands	TechName: Gary Mendel Add: Boeing Avenue 101, 1119pe schipol rijk. Ph: +31 207788200 Email: hostmaster@chello.com

Link Graph

Source of data: alert logs.

MSTREAM DDOS Attacking System Link Graph



This link graph shows the relationship between the Attacker, Handler and Agents of mstream DDOS (Distributed Denial of Service) attacking system. Only the Handler is sitting inside the University's network. Five other systems were RedWorm infected and were connected to the Handler.

Reference:
Dittrich, Dave. “The mstream DDOS attack tool”. 1 May 2000.
URL: http://security.royans.net/info/posts/bugtraq_ddos5.shtml

Method of Analysis

I wrote several scripts to normalize and reorganize the logs to the form that I can manually analyze the contents using spreadsheet and/or command lines..

1. Script scanlog is used to do a preliminary analysis of the network based on the log contents. It tries to locate network devices. It also add the NIC manufacturer name to the report. But you also need to get the NIC OUI file listed below. [20]
2. Ouiconv.txt [21]
3. Script scan_allow2 is used to check for allowed inbound and outbound traffic. But I found that it's not very helpful as the logs file keeps only snort alerts. Other normal traffic that this script relies on are most of the time missing. [22]
4. Script scanalertip generates a summary from the snort generated alert file so that I can have high level view pretty quick. [23]
5. Script scanoos's main job is to reformat the OOS logs as it's original format is not machine friendly. [24]

Linux command lines were used at other time to generate the result I needed. The commands I used were shown in the corresponding sections in this report.

Reference
Besides those listed below, other references are included in the body of this document.

[20] Scanlog
URL: http://members.shaw.ca/kamng/scanlog
§§§§[21] ouiconv.txt
URL: http://members.shaw.ca/kamng/ouiconv.txt.
[22] scan_allow2
URL: http://members.shaw.ca/kamng/scan_allow2
[23] scanalertip
URL: http://members.shaw.ca/kamng/scanalertip
[24] scanoos
URL: http://members.shaw.ca/kamng/scanoos