GIAC PRACTICAL VERSION 4.1

Manuel Humberto Santander Peláez
msantand@eeppm.com
GIAC Computer Forensic Analyst

July 17 2005

TABLE OF CONTENTS

1 PArt one: executive summary

The University has asked for a Security Audit on their IDS system running snort. Three-day log files were downloaded and processed. So far only one signature denotes intrusion but the other ones shows lots of other malicious traffic being transmitted over the University Network, such as:

- RPC Scanning
- IP Spoofing
- Trojan activity

The Snort IDS version that gathered all this alerts is 1.6 – 1.8. This is seen because there's an old ruleset and the most frequent alert detected in the alerts file pertains to that ruleset (http://cvs.sourceforge.net/viewcvs.py/sirt/snortold/RULES.SAMPLE?rev=1.6).

This leads to a conclusion that a Security Model for the Information Technology Infrastructure must be deployed and improved as part of the Security Process on models like ISO17799. The main recommendations for the infrastructure are:

- Disable every unneeded service. Every time you enable an unneeded service or keep default installation of machines, the risk of being attacked when connected to a network increases a lot.
- Perform a regular patch check on every platform. Vulnerabilities are discovered everyday and if they don't get patched as soon as the fix or workaround is available, the risk of being attacked gets increased.
- Keep tuning the IDS for possible false positive alerts. The more the IDS is tuned, the more reliable information delivers to the IT Security area and the response time for an incident goes to the minimum possible.
- Firewall rules and IDS rules are out of date. Many insecure services are allowed to go inside and outside the network. The University has to perform a review of the security policy to enforce them

2 PART TWO: DETAILED ANALYSIS

2.1 Analyzed log files

The IDS logs were downloaded from http://isc.sans.org/logs:

All the alert files were corrupted in some degree, making difficult to be read by a user program. All the numbers shown in figure 1 were gathered after making a correction process to the files. After those corrections the files could be processed.

The rules file of this IDS has activated the spp_portscan preprocessor, making necessary to filter the alert files. Also, dates for OOS files doesn't match the ones shown in the information that they contain, so the OOS files were searched and the ones showed in the tabled matched the dates that the Alert Files and the Scan Files has inside of it.

Figure 1: Analyzed Log Files

Alert Files	Total Lines	Portscan Alerts	Other alerts
http://isc.sans.org/logs/alerts/alert.040219.gz	115038	101803	13235
http://isc.sans.org/logs/alerts/alert.040220.gz	133739	108562	25177
http://isc.sans.org/logs/alerts/alert.040221.gz	182944	139577	43367
OOS Files	**Total of records**		
http://isc.sans.org/logs/oos/oos_report_040215.gz	839		
http://isc.sans.org/logs/oos/oos_report_040216.gz	723		
http://isc.sans.org/logs/oos/oos_report_040217.gz	993		
Scan Files	**Total Lines**		
http://isc.sans.org/logs/scans/scans.040219.gz	1084726		
http://isc.sans.org/logs/scans/scans.040220.gz	1074736		
http://isc.sans.org/logs/scans/scans.040221.gz	1643542		

2.2 Network Topology

A C program was build to process the alert files into a CSV file containing the following fields: Date, Timestamp, Source IP, Source port, Destination IP, Destination port. With this information the information contained in fields Destination IP and Destination port were extracted and then processed to see the port connections for destination hosts. The results are shown in figure 2.

There's lots of buggy traffic on the network, It can be seen IRC bots running on the servers, which are one of the primary source for distributed attacks like distributed denial of service (DDoS), TFTP connections from Internal and External sources, showing unsecured Operating Systems and so on. Detect could have more accurate if raw logs were provided, but OOS files lacks packet data from many alerts, which leaves the port assignment to the IANA Services table and the connection pattern. Those connections that showed they were just scans were discarded.

The alert with the biggest frequency shows lots of RPC connections coming from the outside, about 42.66% of the total alerts. There's a host with seems to be scanning the University network for machines running RPC Services. This can be inferred because the timestamps are very close and sourceports are used in sequence. From the whole sequence, 0.302% of the alerts are catalogued false positives because the sourceport is 80 and the remaining are from hosts 204.152.186.189 (0.015%) and 210.98.224.82. A snip of the log can be seen in figure 3.

Figure 2: Network Topology

Machine	Service
MY.NET.24.47	FTP
MY.NET.24.47	
MY.NET.6.7	
MY.NET.53.29	Helpdesk FTP
MY.NET.70.49	
MY.NET.60.38	SSH
MY.NET.12.6	SMTP
MY.NET.1.3	DNS
MY.NET.1.4	
MY.NET.1.5	
MY.NET.12.217	
MY.NET.12.6	
MY.NET.11.4	WWW
MY.NET.111.72	
MY.NET.112.216	
MY.NET.112.226	
MY.NET.15.202	
MY.NET.15.50	
MY.NET.150.101	
MY.NET.24.34	
MY.NET.24.44	
MY.NET.24.74	
MY.NET.12.4	POP3
MY.NET.11.11	LDAP
MY.NET.11.3	
MY.NET.11.6	
MY.NET.11.7	
MY.NET.30.3	
MY.NET.30.4	
MY.NET.12.7	WWW

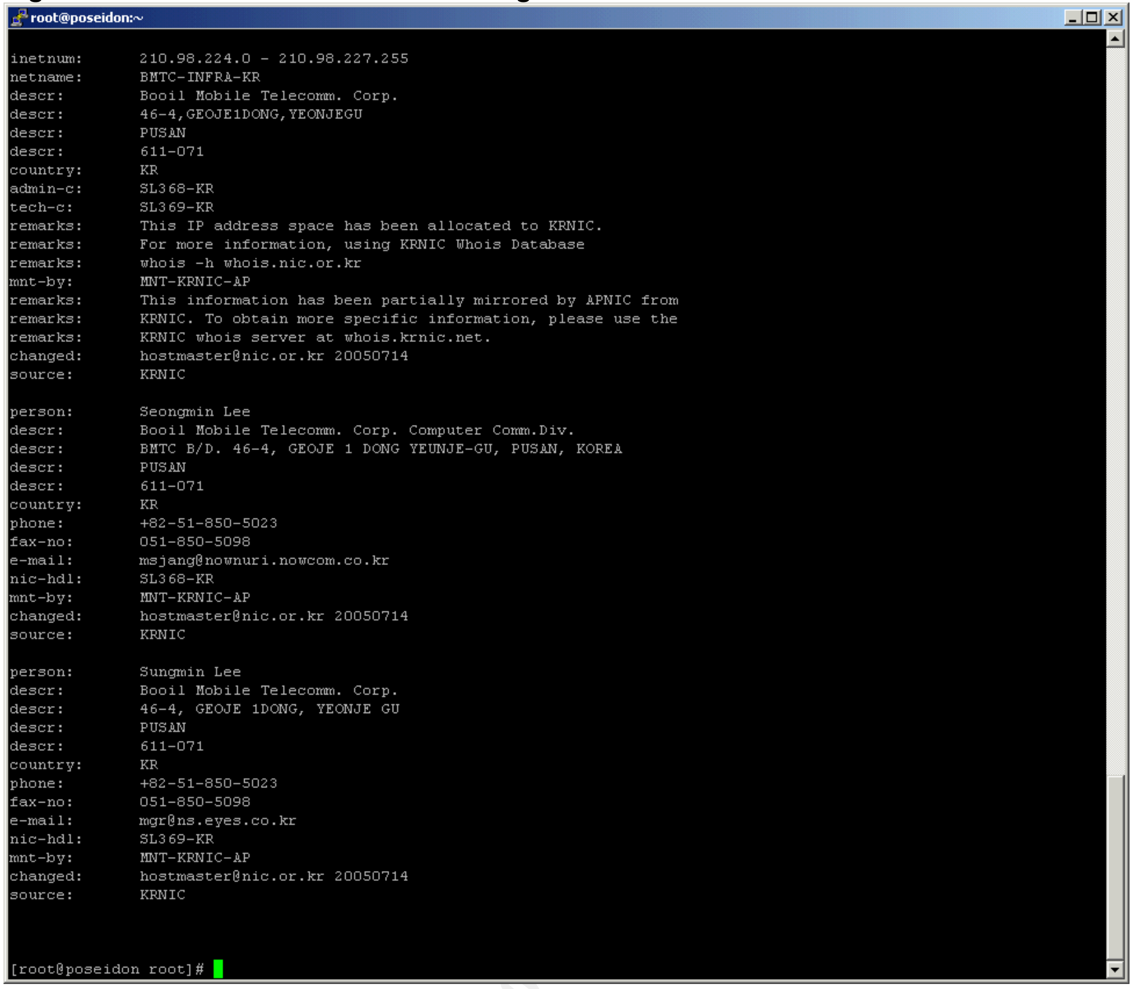
A whois lookup for the IP address was performed and the ip address was matched for the APNIC range, which delegated the usage of it to a Korean Telecom Company. This can be seen on figure 4.

Figure 3: Snip from SUNRPC highport accessscan attempt

Date	Time	Alert	Source IP	Source Port	Destination IP	Destination Port
21-Feb-04	04:35:11 a.m.	SUNRPC highport access!	210.98.224.82	42513	MY.NET.1.97	32771
21-Feb-04	04:35:11 a.m.	SUNRPC highport access!	210.98.224.82	42519	MY.NET.1.103	32771
21-Feb-04	04:35:11 a.m.	SUNRPC highport access!	210.98.224.82	42531	MY.NET.1.115	32771
21-Feb-04	04:35:11 a.m.	SUNRPC highport access!	210.98.224.82	42515	MY.NET.1.99	32771
21-Feb-04	04:35:11 a.m.	SUNRPC highport access!	210.98.224.82	42533	MY.NET.1.117	32771

The logs shows that a RPC scan is being performed on the whole network by the same source IP Address. The traffic flow for this alert is shown in figure 5. Note that the destination ip addresses are multiple and covering whole subnets. Which is why only the subnets are shown..

Figure 4: Whois information for offending IP


```
inetnum:        210.98.224.0 - 210.98.227.255
netname:        BMTC-INFRA-KR
descr:          Booil Mobile Telecomm. Corp.
descr:          46-4,GEOJE1DONG,YEONJEGU
descr:          PUSAN
descr:          611-071
country:        KR
admin-c:        SL368-KR
tech-c:         SL369-KR
remarks:        This IP address space has been allocated to KRNIC.
remarks:        For more information, using KRNIC Whois Database
remarks:        whois -h whois.nic.or.kr
mnt-by:         MNT-KRNIC-AP
remarks:        This information has been partially mirrored by APNIC from
remarks:        KRNIC. To obtain more specific information, please use the
remarks:        KRNIC whois server at whois.krnic.net.
changed:        hostmaster@nic.or.kr 20050714
source:         KRNIC

person:         Seongmin Lee
descr:          Booil Mobile Telecomm. Corp. Computer Comm.Div.
descr:          BMTC B/D. 46-4, GEOJE 1 DONG YEUNJE-GU, PUSAN, KOREA
descr:          PUSAN
descr:          611-071
country:        KR
phone:          +82-51-850-5023
fax-no:         051-850-5098
e-mail:         msjang@nownuri.nowcom.co.kr
nic-hdl:        SL368-KR
mnt-by:         MNT-KRNIC-AP
changed:        hostmaster@nic.or.kr 20050714
source:         KRNIC

person:         Sungmin Lee
descr:          Booil Mobile Telecomm. Corp.
descr:          46-4, GEOJE 1DONG, YEONJE GU
descr:          PUSAN
descr:          611-071
country:        KR
phone:          +82-51-850-5023
fax-no:         051-850-5098
e-mail:         mgr@ns.eyes.co.kr
nic-hdl:        SL369-KR
mnt-by:         MNT-KRNIC-AP
changed:        hostmaster@nic.or.kr 20050714
source:         KRNIC


[root@poseidon root]#
```

2.3 Overview of attacks

Gathering all the alerts, there are 50 unique alerts that were detected and they are shown in figure 6.

2.3.1 Detect 1 – SUNRPC High port access

Description of detect

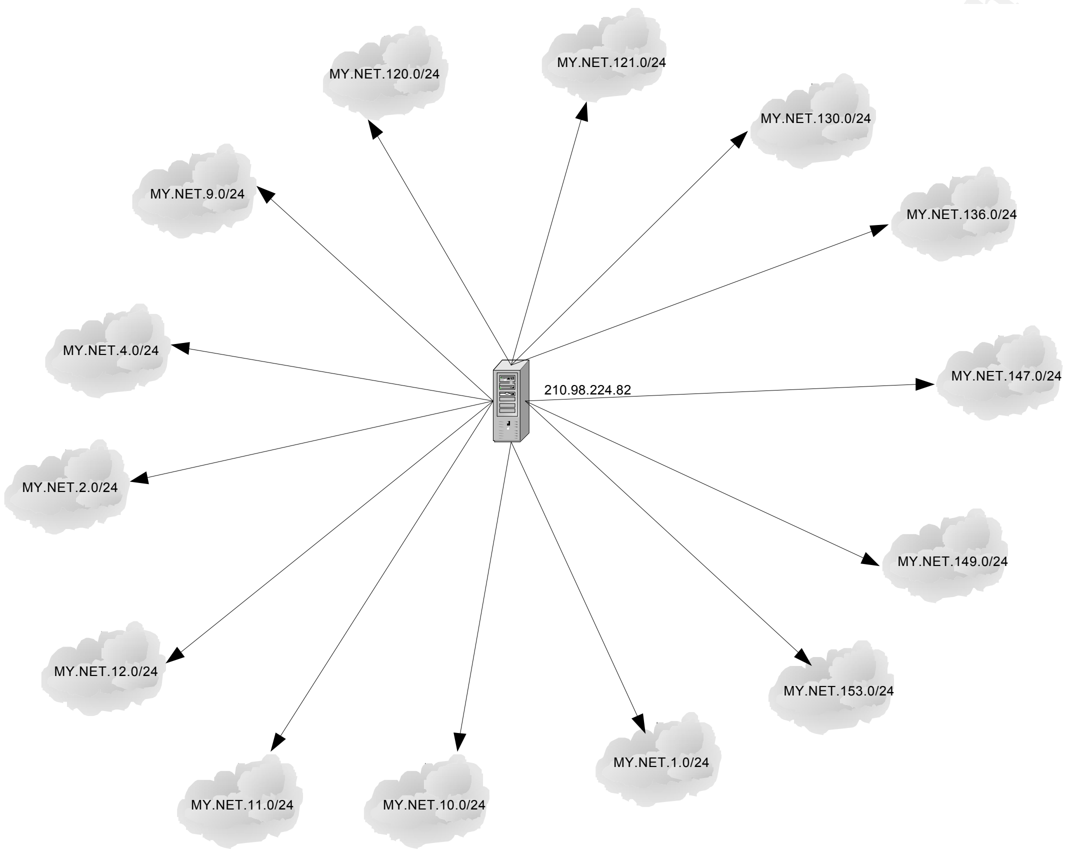
This detect corresponds to a SCAN attempt made over RPC Services over UNIX Machines. This can be seen because all the destination ports are 32771. The goal of this scan is to determine the RPC services that the host is running.

RPC Services are bound to a single program called Portmapper. As example, we can see the Network File System (NFS) using the programs mountd and nfsd bound to the portmapper or rusersd, a program used to query for the logged on users, also bound to the portmapper.

Reason this detect was selected

This attack was selected because it ran all three days from same IP. A search was performed on the Scans files for this scan and no answer was found, only SYN flags in every packet from the external address to all the internal machines.

Figure 5: One to many relationship between the offending host and the subnets scanned



Detect was generated by

No correspondent OOS records for this scan were found, which suggest that the rule which activated the alert has no content checking. This means that every attempt to contact a remote host to UDP port 32771 will trigger this message. Such rule would be:

```
alert ip any any -> $HOME_NET 32771 (msg: "SUNRPC highport access!";)
```

This rule also explains that false positives were detected in the alerts files with sourceport 80, meaning that it detected a HTTP Response to a host under Source Port 32771.

Probability the source address was spoofed

The source ip address might not be spoofed because for three days with pauses of 3 hours the same ip address performed the scan. This doesn't seem to be a denial of service, because the source ports and the IP address increments at same time, about 1 second per increment, which is consistent with RPC scanning on many subnets inside the University Network. Many denial of service and IP spoofing programs pick up random ip address, which doesn't seem to be the case this time. Data from the offending IP can be seen in figure 4.

Figure 6: Unique alertsdetected

Alerts	Occurence
SUNRPC highport access!	34895
TCP SRC and DST outside network	19329
Possible trojan server activity	7612
MY.NET.30.4 activity	7447
SMB Name Wildcard	3477
MY.NET.30.3 activity	3286
Incomplete Packet Fragments Discarded	2814
EXPLOIT x86 NOOP	1234
Null scan!	355
High port 65535 tcp - possible Red Worm - traffic	335
High port 65535 udp - possible Red Worm - traffic	270
NMAP TCP ping!	243
[UMBC NIDS IRC Alert] IRC user /kill detected	97
SMB C access	89
NMAP TCP ping!	52
FTP passwd attempt	36
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	21
EXPLOIT x86 setuid 0	21
Tiny Fragments - Possible Hostile Activity	21
TFTP - Internal TCP connection to external tftp server	19
[UMBC NIDS] External MiMail alert	18
ICMP SRC and DST outside network	17
EXPLOIT x86 setgid 0	16
EXPLOIT x86 stealth noop	12
RFB - Possible WinVNC - 010708-1	8
SYN-FIN scan!	7
connect to 515 from inside	6
TFTP - External TCP connection to internal tftp server	6
TCP	5
DDOS shaft client to handler	4
EXPLOIT NTPDX buffer overflow	3
External FTP to HelpDesk MY.NET.53.29	3
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	2
EXPLOIT FTP passwd retrieval retr path	2
TFTP - Internal UDP connection to external tftp server	2
External FTP to HelpDesk MY.NET.70.49	2
[UMBC NIDS IRC Alert] K\:line'd user detected	1
External FTP to HelpDesk MY.NET.53.29	1
External FTP to HelpDesk MY.NET.70.49	1
Probable NMAP fingerprint attempt	1
TFTP - External UDP connection to internal tftp server	1
Traffic from port 53 to port 123	1
External FTP to HelpDesk MY.NET.70.50	1
[UMBC NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot	1
Attempted Sun RPC high port access	1
DDOS mstream client to handler	1
EXPLOIT x86 NOPS	1
IRC evil - running XDCC	1
RFB - Possible WinVNC - 010708-1	1
Total of alerts	81779

Attack mechanism

This attack was triggered by a scan program. The most common program for performing port scans is nmap, which can be downloaded from http://www.insecure.org. It can be seen on the alerts file that on every day a different network subnet was scanned and in the scans file can be seen that none response was sent from any of the scanned hosts.

Correlations

There's no CVE or advisory on the Internet for this detect, because it doesn't constitutes an attack by itself. This is only a reconnaissance operation performed by the hacker looking for vulnerable RPC services running on the network. This alert is also discussed in *Intrusion Detection (IDS) for wireless networks* – http://www.lurhq.com/idsindepth.html and http://archives.neohapsis.com/archives/snort/2003-3/

Active targeting

There's evidence of scanning almost the whole network. Figure 3 shows a snip of the scan that was targeted to the whole network.

Severity

The formula for calculating severity is the following:

$$Severity = (criticality + lethality) - (systemcountermeasures + networkcountermeasures)$$

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

Criticality is assigned three because there's evidence that the critical servers that host critical services have been scanned also, like DNS Servers.

Lethality is assigned two because this is only a scan and there'd be only a problem of a slow machine if there'd be any response to the attack.

System countermeasures are assigned 5 because there's no evidence of any response for the RPC scan, meaning that those servers don't have the service enabled or they are properly configured.

Network Countermeasures are assigned one because there's evidence that the scan passed inside the network with no filtering.

Recommendations

Since this is only a scanning and an attempt of reconnaissance by a hacker, two recommendations are provided:

- Filter the RPC port on the firewall or place IDS with proper blocking rules so

the risk is not able to materialize.
- Keep disabled RPC on the whole network or place a perimeter security device so only the authorized host are able to talk RPC.

2.3.2 Detect 2 – TCP and SRC outside network

Description of detect

This detect is originated because a packet that doesn't correspond to any subnet inside the University Network was sent from the inside network to the outside. This could mean that an intruder from an external source using maybe a VPN connection or some other link was able to reach the inside network and send packets outside.

This is not what happened, because all occurrences of the alert were triggered by hosts under the address 169.254.0.0/16. According to RFC 3330 which stands all the special ip address under IPV4, those address are special ones assigned for communications between hosts under a single link. Those addresses are obtained when the DHCP server doesn't answer. This is the case of Windows Machines, which uses what ip range to assign addresses when no DHCP was detected.

Reason this detect was selected

This detect was selected because there's a large amount of alerts under this detect and could mean a compromised machine doing IP spoofing to perform Distributed Denial of Service attacks or other kind of damage to other networks on the Internet.

Detect was generated by

No correspondent OOS records were also found for this detect, which suggest that the rule which activated the alert has no content checking. This means that every attempt from any foreign address which is not under the $HOME_NET environment variable to contact a remote external host under any port will trigger this message. Such rule would be:

alert tcp $EXTERNAL_NET any -> $EXTERNAL_NET any (msg: "TCP SRC and DST outside network";)

Those alerts could be false positives, because the IP range match the ones used for IP assigning when no DHCP server answers.

Probability the source address was spoofed

All the IP address under this alert match the 169.254.0.0/16 range and when looked in the scans no info was detected, which could means that the ip communications received from these hosts could be because valid machines under the network weren't able to get an IP address from the DHCP server and then started to try to communicate to the Internet. So, the IP address weren't spoofed.

Attack mechanism

This attack was triggered by several hosts trying to get to the Internet. This means that there are no spoofing filters at the routers or firewalls. In Linux they're called rp_filters, which denies any packet with a source that doesn't match the route table of the machine.

There's something suspicious about the IP address 64.136.21.233. It's a portal whose machine name is my-eap.nyc.untd.com and many machines tried to contact it with invalid address at once. Because there's no content info at the oos files, an inspection on the computers must be performed.

Correlations

There's no CVE or advisory on the Internet for this detect, because it doesn't constitutes an attack by itself. This has been seen under other analysis, like

Active targeting

This detect is neither a scan nor attack, which means that there's no evidence of active targeting. Figure 7 shows this.

Figure 7: Snip from TCP SRC and DST outside network

Date	Time	Alert	Source IP	Source Port	Destination IP	Destination Port
20-Feb	05:06.4	TCP SRC and DST outside network	169.254.204.225	1081	64.136.21.233	80
20-Feb	05:06.6	TCP SRC and DST outside network	169.254.27.223	1368	64.136.21.233	80
20-Feb	05:06.6	TCP SRC and DST outside network	169.254.92.223	1203	64.136.21.233	80
20-Feb	05:06.8	TCP SRC and DST outside network	169.254.49.132	1257	64.136.21.233	80
20-Feb	05:06.8	TCP SRC and DST outside network	169.254.114.132	1092	64.136.21.233	80

Severity

The formula for calculating severity is the following:

$$Severity = (criticality + lethality) - (systemcountermeasures + networkcountermeasures)$$

$$Severity = (1+5) - (1+1) = 4$$

Criticality is assigned one because no critical hosts in the University showed to be compromised.

Lethality is assigned five because if any hosts coud be compromised, there's evidence that external machines could be attacked using IP spoofing which means that attacks like Distributed Denial of Service could be performed.

System countermeasures are assigned one because there's no evidence of any protection on the system that avoids sending spoofed IP packets to the network.

Network Countermeasures are assigned one because there's evidence that ip spoofing packets are able to reach the Internet without any restriction.

Recommendations

Because this detect alerted of no IP spoofing protection on the network, there are two recommendations:

- Place filters on the routers and firewalls that avoids IP packets that doesn't match the route table.
- Place filters, like firewalls, on the servers and user desktops that avoid sending any IP spoofed packets.

2.3.3 Detect 3 – Possible Trojan Server Activity

Description of detect

This detect is originated because a packet with the destination port 27374 was detected. This port is known to be used by Trojans like subseven, Li0n, Ramen, Seeker, The Saint and others. The URL http://www.simovits.com/sve/nyhetsarkiv/1999/nyheter9902.html contains useful port information that Trojan programs uses.

Figure 8: Snip from Possible Trojan Server Activity

Date	Time	Alert	Source IP	Source Port	Destination IP	Destination Port
19-Feb	46:39.8	Possible trojan server activity	MY.NET.84.235	27374	217.225.255.127	4662
19-Feb	46:42.7	Possible trojan server activity	MY.NET.84.235	27374	217.225.255.127	4662
19-Feb	02:19.2	Possible trojan server activity	MY.NET.24.34	80	132.68.1.29	27374
19-Feb	02:19.2	Possible trojan server activity	MY.NET.24.34	80	132.68.1.29	27374
19-Feb	02:19.4	Possible trojan server activity	132.68.1.29	27374	MY.NET.24.34	80
19-Feb	02:19.4	Possible trojan server activity	132.68.1.29	27374	MY.NET.24.34	80
19-Feb	02:19.4	Possible trojan server activity	MY.NET.24.34	80	132.68.1.29	27374

Reason this detect was selected

This detect was selected because a Trojan program is very dangerous. When hackers are not able to break inside networks, they begin to use other techniques like Social Engineering and manage to install under the user desktops programs that allow other computers to take remote control of the machine and as administrator launch other attacks or perform felonies like information theft.

Detect was generated by

No correspondent OOS records were also found for this detects, which suggest that the rule which activated the alert has no content checking. This means that every IP packet that the IDS detect using source or destination port 27374 will trigger this message. Such rule would be:

```
alert tcp any 27374 -> any any (msg: "Possible trojan server activity";)
alert tcp any any -> any 27374 (msg: "Possible trojan server activity";)
```

Note that these rules also will trigger false positives, when source port 27374 gets used by internal or external hosts.

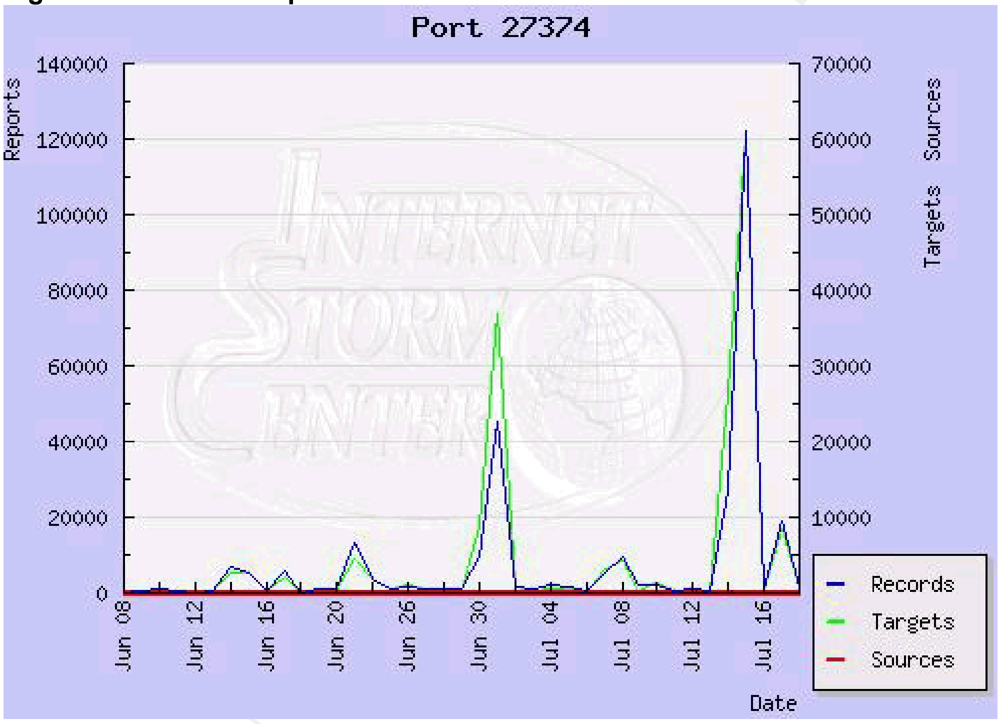
Attack mechanism

This alarm is triggered when a port either is source or destination is set on 27374. Because there's no content checking, every time it appears on the network gets registered, generating lots of false positive alerts.

Correlations

The following is the detection for port 27374 at the Internet Storm Center from SANS Institute:

Figure 5: Detection of port 27374 from Internet Storm Center



Information about Li0n worm can be found at http://www.sans.or/y2k/lion.htm.

Active targeting

When analyzed, this detect shows that only triggered the alarm then the source port 27374 was used, which means that there's no evidence of active targeting.

Severity

The formula for calculating severity is the following:

$$Severity = (criticality + lethality) - (systemcountermeasures + networkcountermeasures)$$

$$Severity = (1+5) - (5+1) = 0$$

Criticality is assigned one because no critical hosts in the University showed to be compromised by a Trojan on this port.

Lethality is assigned five because if any hosts could be compromised, any task could be applied on the machine, which means that attacks like Distributed Denial of Service or compromising any other inside machines could be performed.

System countermeasures are assigned 5 because there's no evidence of any real Trojan traffic on the network.

Network Countermeasures are assigned one because there's evidence that port 27374 is able to pass traffic on both directions with no restrictions.

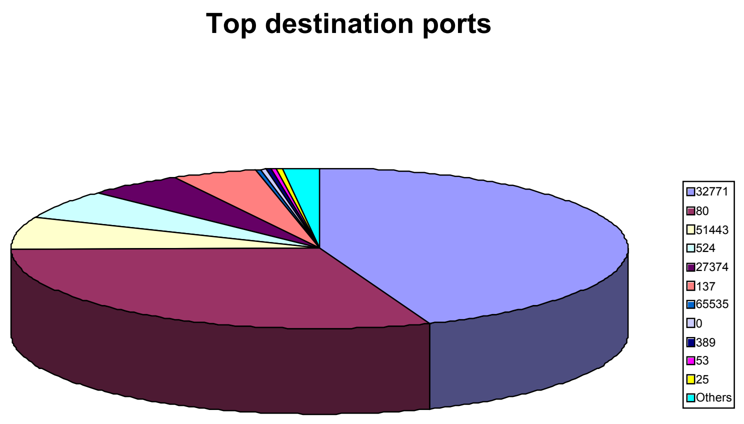
Recommendations

Two recommendations are given:

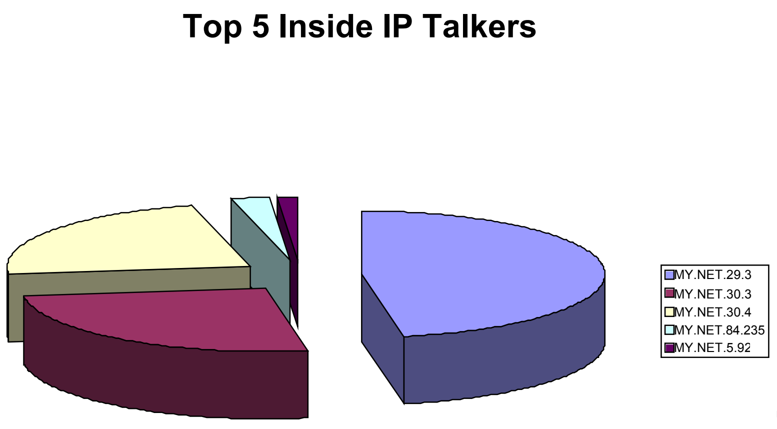
- Place antivirus and antispyware software and run in periodically, so malicious software installed on the machines gets removed.
- Update the rules on the IDS and Firewall to filter any Trojan port using content rules.

2.4 Network Statistics

2.4.1 Five top targeted services



2.4.2 Top 5 inside IP Talkers



2.4.3 Suspicious IP address

First IP address that should be investigated is 210.98.224.82. It's not normal SCAN traffic for three days. This could lead into a really big attack after the reconnaissance phase ends.

Second IP address that should be investigated is 64.136.21.233. It's a portal whose machine name is my-eap.nyc.untd.com. This ip address was tried to be contacted my many machines under the second detect. Because there's no content from this detect on the OOS files, there should be an inspection performed on the desktop machines for possible spyware or virus.

2.5 Correlations from previous students

The following practices were used for correlations:

- Yuan Fan: http://www.giac.org/certified_professionals/practicals/gcia/0850.php.
- Glenn Wtemple: http://www.giac.org/certified_professionals/practicals/gcia/0813.php.
- Michael Gauthier: http://www.giac.org/certified_professionals/practicals/gcia/0811.php.
- Rudy Ristich: http://www.giac.org/certified_professionals/practicals/gcia/0801.php.
- Maarten Van Horenbeek http://www.giac.org/certified_professionals/practicals/gcia/0715.php
- Jason Gordon

http://www.giac.org/certified_professionals/practicals/gcia/0803.php
- Richard Sillito http://www.giac.org/certified_professionals/practicals/gcia/0802.php

2.6 Defensive recommendations

The following recommendations are provided:

- Update firewall rules and IDS signatures: This permit to bring mode protection to the University Network since the most current attacks will be covered and day-zero exploit risk will be minimum.
- Tune the IDS: There are lots of false positive alerts in the logs. If the IDS is tuned, the alerts logged will be really happening and the probabilities for it to be false positive will be minimum, minimizing also the time used to process the information.
- Keep IP logging of the offending packets from the Firewall and IDS detection. This will improve the correlation capabilities of the infrastructure and response time for incident response process.
- Install antivirus and antispyware software: This will erase any Trojan an virus programs installed on the machines and minimize the risk of denial of service and under the network
- Perform a network baseline to the allowed Internet Services: There are malicious services like IRC and Messenger that can be used to perform inside and outside attacks. All services that are no used for the proper activity in the University should be closed and opened by proper justification to the IT Security Officer.
- Perform a Security Assessment for vulnerabilities on the University Network. This can be done with nessus (http://www.nessus.org) and gives really valuable information about the insecure activated services and any patch that might be left for installing.
- Implant a Security policy and also a Security Architecture using known methodologies like ISO17799. This will bring the whole IT infrastructure to a good security level and will be kept improved.
- Implant an implicit deny policy on the firewall. This will keep the integrity on the network as only the proper justified services will be opened and the level risk of the IT infrastructure will be kept minimum.

3 PART THREE: ANALYSIS PROCESS

The alerts were downloaded to a Windows XP Computer using cygwin. The hardware specifications are the following:

- Processor: AMD Sempron 2400 Mhz
- Memory: 1 GB
- Hard Drive: 160 GB

Data was processed with a C program to extract the alerts and putting the alerts into CVS format and then the remaining information was obtained with Microsoft Office Excel. The graphics were developed using Microsoft Office Excel and Microsoft Visio. The network queries were done on a Linux VMWARE on the same windows physical machine.

The steps for data processing were the following:

- A self developed C program was developed to transform the alert files into CVS format. Many errors were found because of what can be a multiple writing snort instance. Data was fixed and then another self developed program was used to tabulate the whole set of alerts. The results for this program are sawn in figure 6.
- Subtotal function from excel was used to get the top 5 IP talkers and top 5 target ports. For manipulating the views of one single alert the filters functionality was used and then the information copied to other worksheets.
- Scan files and OOS files were processed using the grep command from the cygwin program.

4 References

- LURHQ Threat Intelligence Group. *Intrusion Detection In-depth Analysis*. http://www.lurhq.com/idsindepth.html
- SANS Institute. *Lion Worm.* http://www.sans.org/y2k/lion.htm
- Common Vulnerabilites and Exposures. *Search CVE*. http://cve.mitre.org
- Snort Website. *CVS for Ruleset.* http://cvs.sourceforge.net/viewcvs.py/sirt/snortold/RULES.SAMPLE?rev=1.6
- Yuan Fan. *GIAC GCIA Practical*. http://www.giac.org/certified_professionals/practicals/gcia/0850.php.
- Glenn Wtemple. *GIAC GCIA Practical.* http://www.giac.org/certified_professionals/practicals/gcia/0813.php.
- Michael Gauthier. *GIAC GCIA Practical.* http://www.giac.org/certified_professionals/practicals/gcia/0811.php.
- Rudy Ristich. *GIAC GCIA Practical* http://www.giac.org/certified_professionals/practicals/gcia/0801.php.
- Maarten Van Horenbeek. *GIAC GCIA Practical* http://www.giac.org/certified_professionals/practicals/gcia/0715.php

5 APPENDIX

5.1 Appendix 1: Program used to parse the alert file to CVS Format

```
#include <stdio.h>
#include <sys/stat.h>
#include <string.h>

void usage()
{
        printf("\nUso: alertas <archivo>\n");
}



int main(int argc, char *argv[])
{
        int indice,final,contador;
        int i=0,j=0;
        char
nombrearchivoentrada[100]="\0",buffer[400]="\0",res[400]="\0",nombrearchivos
alida[100]="\0";
        if (argv[1] == NULL){
                usage();
                exit(-1);
        }
        strcpy(nombrearchivoentrada,argv[1]);
        strcpy(nombrearchivosalida,nombrearchivoentrada);
        strcat(nombrearchivosalida,".tabulado.csv");
        FILE *entrada=fopen(nombrearchivoentrada,"r");
        FILE *salida=fopen(nombrearchivosalida,"w");
        if (entrada == NULL){
                perror("fopen");
                exit(-2);
        }
        bzero(buffer,400);
        bzero(res,400);
        fgets(buffer,400,entrada);
        while (!feof(entrada)){
                if (!strstr(buffer,"spp_portscan")){
                        indice=final=contador=0;
                        printf("Linea %i y archivo %i\n",i,j);
                        while (buffer[indice] != ' '){
                                if (buffer[indice] == '-')
                                        res[contador]=',';
                                else res[contador]=buffer[indice];
                                contador++;
                                indice++;
                        }
                        res[contador]=',';
                        contador++;
                        indice=indice+7;
                        while  ((buffer[indice]  !=  '[')  ||  (buffer[indice+1]  !=  '*')  ||
(buffer[indice+2] != '*') || (buffer[indice+3] != ']')){
                                res[contador]=buffer[indice];
                                contador++;
                                indice++;
                        }
                        res[contador]=',';
                        contador++;
                        indice=indice+5;
                        while (buffer[indice] != ' '){
                                if (buffer[indice] == ':')
                                        res[contador]=',';
                                else res[contador]=buffer[indice];
                                contador++;
                                indice++;
                        }
                        res[contador]=',';
                        contador++;
                        indice=indice+4;
                        while (buffer[indice] != '\n'){
                              if (buffer[indice] == ':')
                                    res[contador]=',';
                              else res[contador]=buffer[indice];
                              contador++;
                              indice++;
                        }
                        strcat(res,"\n");
                        fputs(res,salida);
                        i++;
                }
                bzero(buffer,400);
                bzero(res,400);
                j++;
                fgets(buffer,400,entrada);
        }
        fclose(entrada);
        fclose(salida);
}
```

5.2 Appendix II: Program used to tabulate the alerts for figure 4

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>


struct nodo {
        char alerta[200];
        int veces;
        struct nodo *siguiente;
        struct nodo *anterior;
};


class tabulacion {
        nodo inicial;
        int ndatos;
public:
        tabulacion();
//      ~tabulacion();
        int comas(char *, int);
        int encontrarenlista(char *);
        void sumar(int );
        void popular(char *);
        void imprimir();
};

void tabulacion::imprimir()
{
        struct nodo *nodos=&inicial;
        char buf[256]="\0";
        FILE *salida=fopen("salida.txt","w");
        fputs("Alerta,Numero de veces\n",salida);
        while (nodos != NULL){
                sprintf(buf,"%s,%i\n",nodos->alerta,nodos->veces);
                fputs(buf,salida);
                bzero(buf,256);
                nodos=nodos->siguiente;
        }
        fclose(salida);
}

int tabulacion::comas(char *buf,int veces)
{
        int contador=0,num=0;
        while ((contador <= strlen(buf)-1) && (num < veces)){
                if (buf[contador] == ',')
```

```
                        num++;
                contador++;
        }
        return contador;
}

int tabulacion::encontrarenlista(char *buf)
{
        int contador=0,encontro=0;
        struct nodo *lista=&inicial;
        while ((lista->siguiente != NULL) && (!encontro)){
                if (!strcmp(buf,lista->alerta))
                        encontro=1;
                lista=lista->siguiente;
                contador++;
        }
        if (encontro)
                return contador-1;
        else
                return -1;
}

void tabulacion::sumar(int posicion)
{
        int contador=0;
        struct nodo *lista=&inicial;
        while (contador != posicion){
                lista=lista->siguiente;
                contador++;
        }
        lista->veces++;
}

void tabulacion::popular(char *buf)
{
        struct nodo *lista=&inicial,*nuevo;
        while (lista->siguiente != NULL)
                lista=lista->siguiente;
        nuevo=new(struct nodo);
        nuevo->siguiente=NULL;
        nuevo->anterior=lista;
        bzero(nuevo, sizeof inicial);
        strcpy(lista->alerta,buf);
        lista->veces=1;
        lista->siguiente=nuevo;
        ndatos++;
}
/*
tabulacion::~tabulacion()
```

```
{
	struct nodo *final=&inicial,*temp;
	while (temp != NULL){
		temp=final->siguiente;
		delete(final);
		final=temp;
	}
}
*/
tabulacion::tabulacion()
{
	bzero(&inicial, sizeof inicial);
	ndatos=0;
	inicial.siguiente=NULL;
	inicial.anterior=NULL;
}

int main(int argc, char *argv[])
{
	tabulacion busqueda;
	char buffer[350]="\0",txtalerta[200]="\0";
	int inicio,fin,indice,nchar,encontro=0;
	FILE *entrada=fopen(argv[1],"r");
	if (entrada == NULL){
		perror("fopen");
		exit(-1);
	}
	fgets(buffer,350,entrada);
	while (!feof(entrada)){
		inicio=busqueda.comas(buffer,2);
		fin=busqueda.comas(buffer,3);
		fin=fin-2;
		bzero(txtalerta,200);
		nchar=0;
		while (inicio <= fin){
			txtalerta[nchar]=buffer[inicio];
			nchar++;
			inicio++;
		}
		encontro=busqueda.encontrarenlista(txtalerta);
		if (encontro != -1)
			busqueda.sumar(encontro);
		else
			busqueda.popular(txtalerta);
		fgets(buffer,350,entrada);
	}
	fclose(entrada);
	busqueda.imprimir();
}
```