



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Network Monitoring and Threat Detection In-Depth (Security 503)"
at <http://www.giac.org/registration/gcia>

GIAC Certified Intrusion Analyst (GCIA) Practical Assignment Version 3.4

DATE OF SUBMISSION: 03/02/2004

© SANS Institute 2004, Author retains full rights.

BY
HITENDRA PATEL

TABLE OF CONTENTS

Table of Contents	2
Assignment 1: Describe state of Intrusion Detection	
ACID (Analysis Console for Intrusion Detection Database) as a Data Mining and Visualization Tool for Snort Intrusion Detection	
Abstract	4
1.0 Introduction	4
1.1 Data mining and Visualization	4
1.2 How can data mining and visualization help in Intrusion Detection?	4
1.3 Open Source Tools Available Today	5
2.0 ACID (Analysis Console for Intrusion Detection Database) for Visualization of Snort Data	6
2.1 ACID Features	6
2.2 Installation	7
2.3 ACID Features for Intrusion Detection	7
2.3.1 Main Summary Page	7
2.3.2 Listing Protocol Data	8
2.3.3 Payload Details	8
2.3.4 Unique IP Link Page	9
2.3.5 Incident Storage and Reporting	10
2.3.6 Searching	10
3.0 Practical Considerations to make System Efficient	11
3.1 Some Limitation Of ACID	11
4.0 Summary	11
5.0 Reference	12
Assignment: 2 Network Detects	
First Detect: FTP SITE EXEC	
1.0 Source of Trace	13
2.0 Detect was Generated By	14
3.0 Probability the Source Address was Spoofed	16
4.0 Description of Attack	17
5.0 Attack Mechanism	17
6.0 Correlations	20
7.0 Evidence of Targeting	20
8.0 Severity	21
9.0 Defensive Recommendation	21
10.0 Multiple Choice Question	22
Second Detect: Port 0 Traffics	
1.0 Source of Trace	22
2.0 Detect was Generated By	23

3.0 Probability the Source Address was Spoofed	24
4.0 Description of Attack	25
5.0 Attack Mechanism	27
6.0 Correlations	28
7.0 Evidence of Targeting	28
8.0 Severity	28
9.0 Defensive Recommendation	29
10.0 Multiple Choice Question	29

Third Detect: UDP Scan by ISS

1.0 Source of Trace	31
2.0 Detect was Generated By	33
3.0 Probability the Source Address was Spoofed	36
4.0 Description of Attack	37
5.0 Attack Mechanism	37
6.0 Correlations	39
7.0 Evidence of Targeting	39
8.0 Severity	39
9.0 Defensive Recommendation	40
10.0 Multiple Choice Question	40

Assignment: 3 Analyze This!

Executive Summary	41
Logs Analyzed	42
Summary of Alerts	42
Frequent Alert Details	46
Scan Analysis	53
OOS Log Analysis	57
Conclusion and Recommendations	60
Process Used in Analysis	60
References	61

APPENDIX: A Link Graph

APPENDIX: B IP Info for Top Source Host in Alert Logs

APPENDIX: C Scan Analysis

©

ASSIGNMENT: 1: Describe the State Of Intrusion Detection ACID (Analysis Console for Intrusion Detection Database) as a Data Mining and Visualization Tool for Snort Intrusion Detection

Abstract

Intrusion detection is an essential component of security administration in modern world. In any busy network Intrusion Detection systems will generate a large amount of events, which makes impossible for human analyst to review. This paper will discuss ACID as a data mining and visualization tool, which can help the analyst to reduce some burden by summarizing and classifying events in a logical format.

1.0 Introduction

Today with explosive growth of the Internet and increasing availability of attacking tools, intrusion detection becomes a critical part of security administration. The Intrusion detection systems collect and analyze network activity data to determine whether there is an attack occurring. There are two classification of analysis: misuse detection and anomaly detection. Misuse detection uses a known pattern of attack called signatures. Misuse detection is not effective against unknown attacks. In anomaly detection, the System defines the normal behavior in advance; which is known as a profile. Any deviation from normal behavior is then reported as a potential attack. The strength of anomaly detection is its ability to detect unknown attacks but prone to falsely identifying events as an attack, resulting false alarm. In modern intrusion detection systems it is important to combine both of these approaches.

1.1 Data mining and Visualization

According to ^[11] Steven Noel in “Modern Intrusion Detection, Data mining and Degree of attack Guilt”, data mining refers to a process of nontrivial extraction of implicit, previously unknown, and potentially useful information from a database. Visualization allows the analyst to see and comprehend large amounts of complex data in a short period for review.

Intrusion Detection Data mining is a process where large sets of previously collected data is filtered, transformed, and organized into information sets. This information can be used by an analyst to find out hidden undetected attacks.

1.2 How can data mining and visualization help in intrusion detection?

Medium to large organizations are still subjects to constant attack by outsiders. With the progress of technology and network speeds (i.e. bandwidth) increasing IDS sensors can easily generate a large number of events. Any un-tuned Intrusion detection

sensor can produce thousands of events, so there is a risk of the console overwhelming the analyst with the false positive events, giving them no opportunity to focus on relatively few events of real interest. There is a chance attack may be directed to a user interface which will prevent the analyst from noticing some smaller number of serious events. With a signature database known to the attacker, it is no doubt feasible for an attacker to implement or design a tool, which triggers each possible signature on a monitor network, thus the attacker can fill out a top-level signature based display.

A few specific things that data mining can contribute are the remove of normal activity from alarms data to allow the analyst to focus on a real attack, identify false alarms (generated by bad signatures), find anomalous activity and identify ongoing patterns. The data mining is not used to replace a human analyst but to help them identify significant events and reduce times wasted with false positive events. An example of data flow in the MITRE network is shown in figure 1.0^[3].

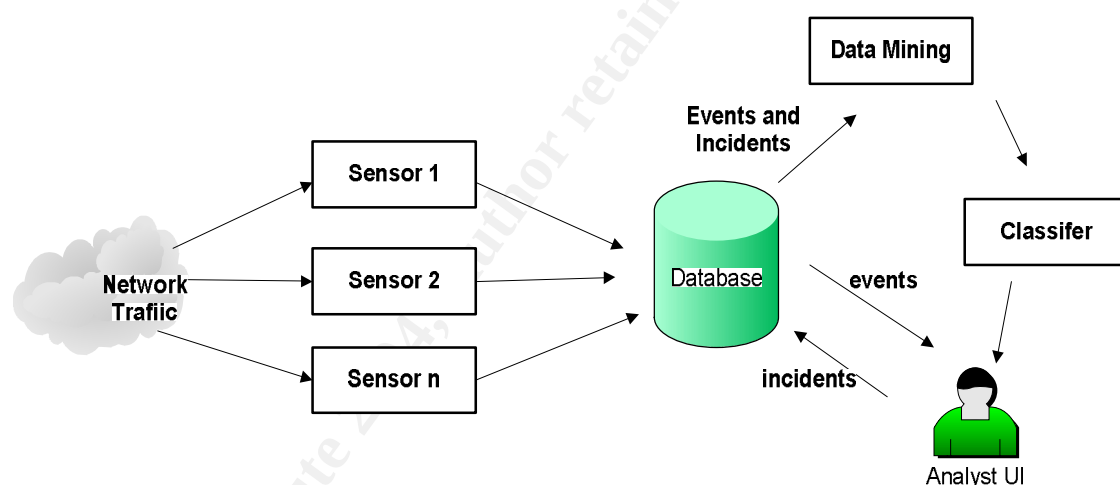


Figure 1.0 Data flow in the MITRE network

Data Mining can be accomplished with Data summarization, visualization, aggregation and classification. Data mining can be used to identify recurring sequences of alarms, identify most relevant data and provide the analyst with different “views” of data to aid in analysis. Without automated support this task is difficult to accomplish due to the amount of events. Visualization gives the ability to cue the analyst with the use of color, shape or patterns.

1.3 Open Source Tools Available Today

Snort is an open-source network Intrusion detection system that is in wide use today. As of this writing, there are 2108 signatures in the standard signature database for known attacks. In the past few years Snort has been improved with support for

defragmentation, TCP re-assembly, and anomaly detection with spade from Silicon Defense. On any busy network connected to the Internet, a default install of Snort can easily produce thousands of false alarms a day. It is necessary to fine tune Snort for your appropriate needs for better performance. Directly examining the logs of all alerts is inconvenient and cumbersome.

There are many open-source tools available today to visualize and mining Snort alert data in the database (i.e. MySQL, and MS-SQL) such as ACID (Analysis Console for Intrusion Database), Snortsnarf, Snort Report and SnortPHP. All these tools are allowing the analyst to analyze and present Snort data in a web interface. For this paper we have selected ACID for discussion.

2.0 ACID (Analysis Console for Intrusion Detection Database) for Visualization of Snort Data

ACID is a PHP-based engine, allowing the analyst to search and analyze the security incident database. ACID was developed by Roman Danyliw at the CERT Coordination Center and was initially used as an Aircert project. The latest version of ACID is available from <http://www.cert.org/kb/acid/>. ACID is still open-source under GPL licensing.

In order to support multiple database types ADODB is required. The reason behind this is PHP does not have any common database API for accessing multiple databases. In order to support multiple types of database some sort of database abstraction is required. As of today PHP has support for MySQL, postgresSQL and MS-SQL.

2.1 ACID Features

ACID offers many features as described below:

1. Searching can be done on many criteria such as source and destination address, time, ports etc.
2. Graphically display the different header part as well as payload information. (Layer-3 and Layer-4 packet information)
3. Alerts can be logically grouped to create an incident report, exporting and deleting and sending to a specified email address.
4. Graphical and statistical representation of alert data based on time, sensor, signature, protocol, IP Address, TCP/UDP ports, or Classification.
5. Snapshots can be taken of alert data which will let the analyst view alerts for the last 24 hours, unique alerts and so on.
6. Allow the analyst to go to external whois database to search for IP address information. (i.e. ARIN, Samspace etc.)

We will explain this in detail in later sections.

2.2 Installation

ACID needs other packages to install such as PHP, GD, PHPLOTT and so on. Installation is out of scope of our discussion in this paper but you can find many documents from the official site <http://www.cert.org/kb/acid/> or refer to this book: “Intrusion Detection Systems with Snort” by Rafieeq Ur Rehman. In this book he has explained a step-by-step installation of ACID, MySQL and Snort in detail.

2.3 ACID Features for Intrusion Detection

2.3.1 Main Summary Page

The very first page an analyst looks at when viewing ACID is the main summary page. An example of this is presented in Figure 2.0.

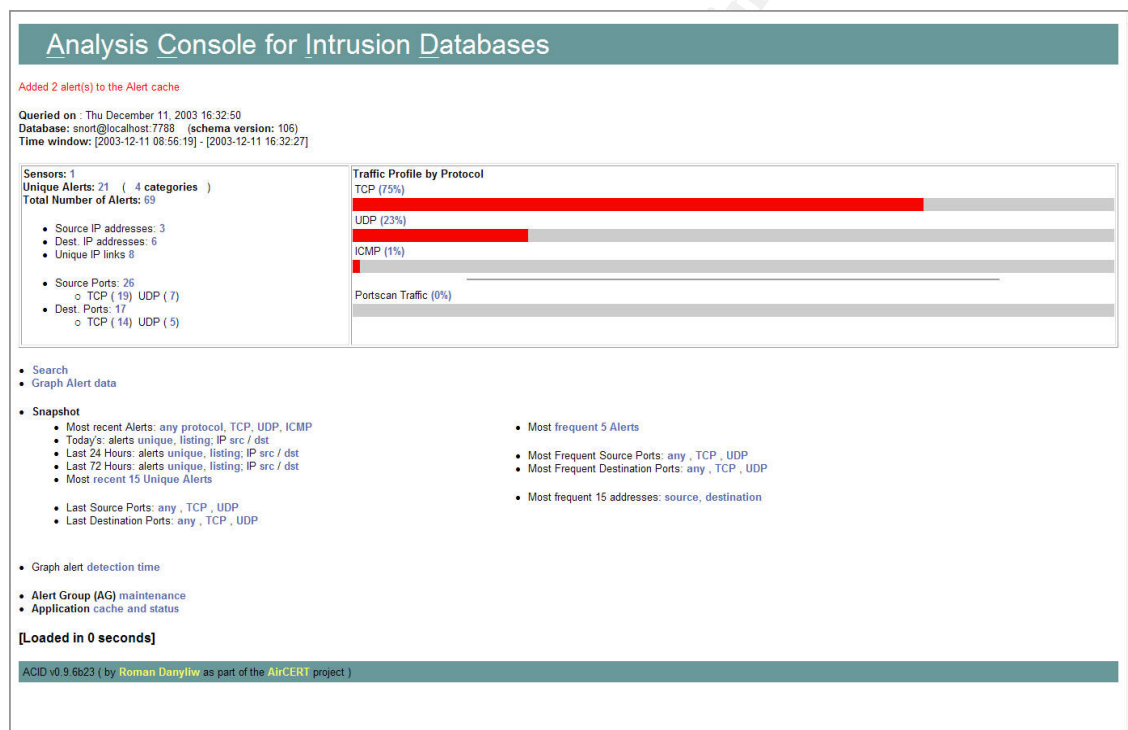


Figure 2.0 Main summary page

This provides top-level overview of all alert data. It has a different section to display different information. The analyst can view traffic profile by protocols. This page gives a great deal of information such as a list of sensors, number of unique alerts, total number of alerts generated, breakdown alert by source IP or Destination IP or by ports.

2.3.2 Listing Protocol Data

From the main page, the analyst can click on protocol data to get more information about packets logged for that protocol. The analyst can also view unique or total alerts generated by Snort signatures. An example of this is presented in figure 3.0. The table of signatures contains text of the signature with a link of a website (i.e. Snort, CVE) with more information about the signature, classification, total number of alerts, the number of distinct source and destination IP address, and the time frame of first and last alerts. This information helps an analyst decide which alert to investigate. This data also helps tuning the IDS sensor for false alarms generated by bad signatures, which assist in isolating, investigating and prioritizing events. The analyst can select some specific events, which require further investigation and add them to logical AG group or email to another security agency for incident reporting. ACID has functionality to manage all events, such as allowing the analyst to delete false alerts from the database or move, copy or archive alerts in the database for storage maintenance. The analyst can dig further alerts by each signature that lists a distinct source and destination IP address.

ACID Alert Listing

Home Search | AG Maintenance [Back]

Added 0 alert(s) to the Alert cache

Queried DB on: Thu December 11, 2003 16:33:31

Meta Criteria: any
IP Criteria: any
Layer 4 Criteria: none
Payload Criteria: any

Displaying alerts 1-21 of 21 total

	< Signature >	< Classification >	< Total # >	Sensor #	< Src. Addr. >	< Dest. Addr. >	< First >	< Last >
☐	[snort] (spp_portscan2) Portscan detected from 10.10.1.68: 6 targets 6 ports in 1 seconds	unclassified	2 (3%)	1	1	1	2003-12-11 08:59:20	2003-12-11 11:19:20
☐	[snort] (spp_portscan2) Portscan detected from 10.10.1.68: 6 targets 6 ports in 0 seconds	unclassified	1 (1%)	1	1	1	2003-12-11 09:09:19	2003-12-11 09:09:19
☐	[snort] (spp_portscan2) Portscan detected from 10.10.1.125: 6 targets 10 ports in 54 seconds	unclassified	2 (3%)	1	1	2	2003-12-11 08:56:19	2003-12-11 09:09:20
☐	[arachnIDS][snort] ICMP L3retreiver Ping	attempted-recon	1 (1%)	1	1	1	2003-12-11 13:47:53	2003-12-11 13:47:53
☐	[snort] NETBIOS SMB winreg access (unicode)	attempted-recon	1 (1%)	1	1	1	2003-12-11 13:47:53	2003-12-11 13:47:53
☐	[snort] (spp_portscan2) Portscan detected from 10.10.2.55: 1 targets 21 ports in 24 seconds	unclassified	1 (1%)	1	1	1	2003-12-11 13:47:53	2003-12-11 13:47:53
☐	[cve][icat][cve][icat][snort] SNMP request tcp	attempted-recon	6 (9%)	1	1	1	2003-12-11 13:47:54	2003-12-11 13:50:54
☐	[cve][icat][cve][icat][snort] SNMP trap tcp	attempted-recon	6 (9%)	1	1	1	2003-12-11 13:47:54	2003-12-11 13:50:54
☐	[cve][icat][cve][icat][snort] SNMP AgentX/tcp request	attempted-recon	6 (9%)	1	1	1	2003-12-11 13:47:54	2003-12-11 13:50:54
☐	url[snort] SCAN SOCKS Proxy attempt	attempted-recon	6 (9%)	1	1	1	2003-12-11 13:47:54	2003-12-11 13:50:54
☐	[snort] SCAN Squid Proxy attempt	attempted-recon	6 (9%)	1	1	1	2003-12-11 13:47:54	2003-12-11 13:50:55
☐	[snort] SCAN Proxy (8080) attempt	attempted-recon	6 (9%)	1	1	1	2003-12-11 13:47:55	2003-12-11 13:50:55
☐	[cve][icat][arachnIDS][snort] DDOS mstream client to handler	attempted-dos	9 (13%)	1	1	1	2003-12-11 13:47:55	2003-12-11 13:50:55
☐	[snort] BACKDOOR DeepThroat 3.1 Connection attempt	misc-activity	2 (3%)	1	1	1	2003-12-11 13:48:05	2003-12-11 13:48:05
☐	[snort] BACKDOOR DeepThroat 3.1 Connection attempt [3150]	misc-activity	2 (3%)	1	1	1	2003-12-11 13:48:05	2003-12-11 13:48:05
☐	[snort] (spp_portscan2) Portscan detected from 10.10.2.55: 1 targets 21 ports in 31 seconds	unclassified	1 (1%)	1	1	1	2003-12-11 13:48:56	2003-12-11 13:48:56
☐	[cve][icat][cve][icat][snort] SNMP request udp	attempted-recon	4 (6%)	1	1	1	2003-12-11 13:48:56	2003-12-11 13:51:11
☐	[cve][icat][cve][icat][snort] SNMP trap udp	attempted-recon	4 (6%)	1	1	1	2003-12-11 13:48:56	2003-12-11 13:51:11
☐	[snort] (spp_portscan2) Portscan detected from 10.10.2.55: 1 targets 21 ports in 26 seconds	unclassified	1 (1%)	1	1	1	2003-12-11 13:50:53	2003-12-11 13:50:53
☐	[snort] (spp_portscan2) Portscan detected from 10.10.1.125: 6 targets 13 ports in 13 seconds	unclassified	1 (1%)	1	1	1	2003-12-11 16:31:37	2003-12-11 16:31:37
☐	[snort] (spp_portscan2) Portscan detected from 10.10.1.125: 6 targets 11 ports in 25 seconds	unclassified	1 (1%)	1	1	1	2003-12-11 16:32:27	2003-12-11 16:32:27

Action: { action } Selected ALL on Screen

[Loaded in 0 seconds]

ACID v0.9.6b23 (by Roman Danyliw as part of the AirCERT project)

Figure 3.0 Unique Alert Listing

2.3.3 Payload Details

This page shows layer-3 and layer-4 packet information. Figure 4.0 shows details about a particular packet that the analyst would see when they click on an alert on a query page. The topmost part shows general details, which includes the signature so

that the analyst can easily determine the circumstances under which the signature alert was generated. The payload is displayed both in hexadecimal and ASCII text. A different color header makes it very easy to understand visually. At the bottom of the payload data navigation buttons are provided that can be used to move to next and previous alerts quickly.

Meta	ID #	Time	Triggered Signature									
	1 - 19	2003-12-11 13:47:53	[snort] NETBIOS SMB winreg access (unicode)									
	Sensor	name		interface								
	SALES-002	:DeviceNPF_{1FAF0A67-72C3-4731-AE58-7C91C7CCE51C}		DeviceNPF_{1FAF0A67-72C3-4731-AE58-7C91C7CCE51C}								
Alert Group	ID #	Name	Description									
	1	incident1	first attack seen									
IP	source addr	dest addr	Ver	Hdr Len	TOS	length	ID	flags	offset	TTL	chksum	
	10.10.2	10.10.9	4	5	0	144	60625	0	0	128	60925	
	FQDN	Source Name	Dest. Name									
	IS-001	SALES-002										
	Options none											
TCP	source port	dest port	R1	R0	U	A	P	S	R	S	F	I
	4325	139			X	X						
	seq #	ack	offset	res	window	urp	chksum					
	1065963415	611126866	5	0	63978	0	2473					
	Options none											
Payload	length = 104											
	000 : 00 00 00 64 FF 53 4D 42 A2 00 00 00 00 18 07 C8 ...d.SMB.....											
	010 : 00 00 00 00 00 00 00 00 00 00 00 00 08 A4 09 ...0.....											
	020 : 00 08 30 00 18 FF 00 DE DE 00 0E 00 16 00 00 00 ...0.....											
	030 : 00 00 00 00 9F 01 02 00 00 00 00 00 00 00 00 00 ...0.....											
	040 : 00 00 00 00 03 00 00 00 01 00 00 00 40 00 00 00 ...0.....											
	050 : 02 00 00 00 03 11 00 00 5C 00 77 00 69 00 6E 00 ...N..w.i.n.. r.e.g...											
060 : 72 00 65 00 67 00 00 00												
<< Previous #17-(1-30) >> Next #19-(1-32)												

Figure 4.0 Payload Data

2.3.4 Unique IP Link Page

This page helps the analyst to view the overall picture of all events about a source generating alert and destination with fully qualified domain names, protocol, total number of events generated from the same source, unique events, etc. An example of this is presented in Figure 5.0

ACID

IP Links

Home

Search

AG Maintenance

Added 0 alert(s) to the Alert cache

Queried DB on : Fri December 12, 2003 12:31:46

Meta Criteria

any

IP Criteria

any

Layer 4 Criteria

none

Payload Criteria

any

Displaying alerts 1-19 of 19 total

Source FQDN	< Source IP >	Direction	< Destination IP >	Destination FQDN	Protocol	Unique Dst Ports	Unique Events	Total Events
IS-001	10.10.	-->	10.10.9	SALES-002	TCP	10	11	76
snv-sys	10.10.1.	-->	10.10.8.	DEV-RE	TCP	1	1	1
snv-sys	10.10.	-->	10.10.1.	snv-mb	TCP	1	1	1
snv-sys	10.10.1.	-->	10.10.9.	SALES-001	TCP	1	1	1
overwatch	10.10.	-->	10.10.1.	Unable to resolve address	UDP	1	2	14
snv-sys	10.10.1.	-->	10.10.8.	QA-VP	TCP	1	1	1
IS-001	10.10.	-->	10.10.9	SALES-002	UDP	5	5	17
IS-0015	10.10.2.	-->	10.10.9.	SALES-002	ICMP	0	1	2
overwatch	10.10.	-->	10.10.1	Unable to resolve address	UDP	1	1	1
snv-sys	10.10.1.	-->	10.10.	QA-002	TCP	1	1	1
snv-sys	10.10.1.	-->	10.10.9.	DEV-002	TCP	1	1	1
snv-sys	10.10.1.	-->	10.10.	CORP-SC	TCP	1	1	1
snv-syslog	10.10.1	-->	10.10.8.	TECH-002	TCP	1	1	1

Back

Figure 5.0 IP Links

This page is an example of what was really useful during “nachi” and “code red” worm attacks. An analyst can easily find the source of the problem and directions where traffic is flowing.

2.3.5 Incident Storage and Reporting

ACID has a feature to store and logically group events for incident reporting with certain label data. Alerts of identified interest are stored in this manner. During any incident all alerts with label data will help the analyst or incident handler with further documentation, which can be used and prepared for producing reports for management. Figure 6.0 shows example of logical grouping.

ACID

Alert Group (AG) Maintenance

Home
Search | AG Maintenance

[Back]

list all | create | view | edit | delete | clear

List groups

ID	Name	# Alerts	Description	Actions
1	incident1	9	first attack seen	edit delete clear
2	for expert review	22	need further investigation	edit delete clear

[Loaded in 0 seconds]

ACID v0.9.6b23 (by Roman Danyliw as part of the AirCERT project)

Figure 6.0 Logical grouping for reporting

As shown in figure 6.0 this is concise way to organize all interested alerts for detailed review. This even allows groups to receive alerts via email in summary or full, including any outside experts or agency for expert analysis and can also be archived for historic documents in the database.

2.3.6 Searching

One important feature of ACID is that it can be used to search alert data by sensor, signature, source IP address, destination or time of alerts. All this functionality is

available from the Main page. As of this writing ACID has the capability of searching by Layer-4 TCP/IP options and payload criteria. Searching for alerts is made very easy. All criteria that the analyst specifies in the search screen are translated to a SQL statement that is passed to a backend database server so query results will display faster.

3.0 Practical Considerations to make System Efficient

Due to the high volume of data and frequency of input, regular storage maintenance is required for any database. Maintaining a reliable size of the alert database is important for performance. One informative performance benchmarking paper is available at the ACID official site <http://www.cert.org/kb/acid/> for MySQL vs. PostgreSQL. As the size of the alert database increases you will see some poor performance loading pages or searching any alerts. There are no rules for database size; it is dependent upon many parameters such as hardware platform, memory etc. ACID has some features available for maintenance. The analyst can move or copy alerts to a different database for archiving according to organization policy for historic documents.

As explained earlier any default install of Snort can generate thousands of alerts. An optimization of the IDS sensor is necessary for any bad Snort signatures. The analyst should turn off bad signatures, which produce false positive or negative alerts and unnecessary preprocessor activity.

3.1 Some Limitation Of ACID

In general ACID is a good product for Data mining and visualizing Snort alert data but is still missing some functionality that any security profession may like. ACID has a web link for alerts for IP lookup such as whois and Samspade, but having functionality for Dig and Nmap would be beneficial as well. NTOP integration probably will help for anomaly detection. Centralized correlation of all data is also important to find out about when the attack is actually happening. There is also a need for some further enhancement in the graph capability.

4.0 Summary

The main benefit of ACID is the ability to efficiently manage the large number of events by efficiently ignoring false alarms that Snort typically produces when it has been installed on any busy network. This is especially true when a new sensor has just been installed on the network without tuning for any rule set or preprocessor. In this situation, ACID is a very powerful tool for visualizing Snort alerts because it segregates all events of a particular kind together; separately from uninterested events. Thus the analyst can pay attention to the few very important alerts.

The central point of this paper is that ACID is a one of the powerful tools for data mining and the visualization of Snort alert data. By carefully and adaptively dividing up the alerts into a hierarchy of smaller groups, one can guarantee that the analyst only has to review the important events of his choice. Data mining products are not used to replace the human analyst, but to reduce the burden of their task by allowing the analyst to use experience on those alarms, which are likely to create a much bigger problem.

5.0 REFERENCES

- 1) Aleksandar Lazarevic, Jaideep Srivastava, and Vipin Kumar, AHPCRC–UM, “Data Mining Techniques for Network Intrusion Detection”,
<http://www.ahpcrc.org/publications/archives/v12n2/Story1/>
- 2) Daniel Barbara, “ Application of Data Mining to Intrusion Detection”,
<http://www.isse.gmu.edu/~csis/faculty/barbara.pdf>
- 3) Dr. Eric Bloedorn, the MITRE Corporation, “Data Mining for Improving Intrusion Detection”,
http://www.mitre.org/work/tech_papers/tech_papers_00/bloedorn_datamining/index.html
- 4) Eric Bloedorn, Alan D. Christiansen, William Hill, Clement Skorupka, Lisa M. Talbot, Jonathan Tivel, The MITRE Corporation, “Data Mining for Network Intrusion Detection: How to Get Started”,
http://www.mitre.org/work/tech_papers/tech_papers_01/bloedorn_datamining/index.html
- 5) <http://acidlab.sourceforge.net/>
- 6) Jesus Mena, “Investigative Data Mining for Security and Criminal Detection, First Edition”, Butterworth-Heinemann; 1 edition (January 31, 2003)
- 7) Klaus Julisch, “ Data Mining For Intrusion Detection, A critical Review”,
<http://www.zurich.ibm.com/~kju/excerpt.pdf>
- 8) Manh Phung (2000),” Data Mining in Intrusion Detection”,
http://www.sans.org/resources/idfaq/data_mining.php
- 9) Project,” A Data Mining Approach for Building Cost-sensitive and Light Intrusion Detection Models, <http://www.cc.gatech.edu/~wenke/project/id.html>
- 10) Rafeeq ur Rehman, “Intrusion Detection with Snort, Apache, MySQL, PHP, Prentice Hall PTR; 1st edition (May 8, 2003)
- 11) Steven Noel, Duminda Wijesekera, and Charles Youman, “Modern Intrusion Detection, Data Mining, And Degrees of Attack Guilt”,
<http://www.ise.gmu.edu/~snoel/IDS%20Chapter.htm>
- 12) Steven Noel, “Modern Intrusion Detection, Data mining, and Degrees of Attack guilt”, <http://www.isse.gmu.edu/~snoel/IDS%20chapter.pdf>
- 13) Steven Noel, “Data Mining for Intrusion Detection”,
<http://www.isse.gmu.edu/~snoel/ID%20data%20mining.htm>
- 14) Wenke Lee and Salvatore J. Stolfo,” Data Mining Approaches for Intrusion Detection”, <http://www1.cs.columbia.edu/~sal/hpapers/USENIX/usenix.html>
- 15) William Yurcik, Kiran Lakkaraju, “A Prototype Tool for Visual Data Mining of Mining of Network Traffic for Detection”,
<http://www.ncsa.uiuc.edu/People/jbarlow/publications/ICDM-DMSEC03.pdf>
- 16) Wenke Lee,” Real Time Data Mining-based Intrusion Detection”,
<http://www1.cs.columbia.edu/ids/concept/>

DETECT: 1 FTP SITE EXEC & ATTACK-RESPONSES ID Check Returned Root

(Posted on: Thu 1/22/2004)

1) SOURCE OF TRACE:

The Raw tcpdump logs were obtained from <http://www.incident.org/logs/raw> website. The archive file 2003.12.15.tgz contained several days' worth of log files, but 2003.12.15.12 is used for this analysis. The IP addresses shown in log file have been changed to hide true identity.

By using windump, I have learned information at the IP level about the unknown network.

windump -r c:\gcia\2003.12.15.12 -e -v host 10.10.10.228 and host 172.20.201.135 and (tcp port 35886 and tcp port 21)

```
14:21:22.084529 0:3:ff:df:95:84 0:50:56:40:0:6d ip 74: IP (tos 0x0, ttl 64, id 17676, len 60)
10.10.10.228.35886 > 172.20.201.135.21: S [tcp sum ok] 2614831172:2614831172(0) win 5840 <mss
1460,sackOK,timestamp 355656 0,nop,wscale 0> (DF)
14:21:22.087897 0:50:56:40:0:6d 0:3:ff:df:95:84 ip 74: IP (tos 0x0, ttl 62, id 43249, len 60)
172.20.201.135.21 > 10.10.10.228.35886: S [tcp sum ok] 1710593240:1710593240(0) ack 2614831173
win 32120 <mss 1460,sackOK,timestamp 2547417 355656,nop,wscale 0> (DF)
14:21:22.088966 0:3:ff:df:95:84 0:50:56:40:0:6d ip 66: IP (tos 0x0, ttl 64, id 17677, len 52)
10.10.10.228.35886 > 172.20.201.135.21: . [tcp sum ok] ack 1 win 5840 <nop,nop,timestamp 355657
2547417> (DF)
14:21:22.163708 0:50:56:40:0:6d 0:3:ff:df:95:84 ip 171: IP (tos 0x10, ttl 62, id 43256, len 157)
172.20.201.135.21 > 10.10.10.228.35886: P 1:106(105) ack 1 win 32120 <nop,nop,timestamp 2547425
355657> (DF)
14:21:22.163708 0:3:ff:df:95:84 0:50:56:40:0:6d ip 66: IP (tos 0x0, ttl 64, id 17678, len 52)
10.10.10.228.35886 > 172.20.201.135.21: . [tcp sum ok] ack 106 win 5840 <nop,nop,timestamp 355663
2547425> (DF)
14:21:22.170999 0:3:ff:df:95:84 0:50:56:40:0:6d ip 75: IP (tos 0x0, ttl 64, id 17679, len 61)
10.10.10.228.35886 > 172.20.201.135.21: P [tcp sum ok] 1:10(9) ack 106 win 5840 <nop,nop,timestamp
355664 2547425> (DF)
....
14:21:30.222374 0:3:ff:df:95:84 0:50:56:40:0:6d ip 75: IP (tos 0x0, ttl 64, id 17788, len 61)
10.10.10.228.35886 > 172.20.201.135.21: P [tcp sum ok] 16643:16652(9) ack 34802 win 20874
<nop,nop,timestamp 356444 2548019> (DF)
14:21:30.237991 0:50:56:40:0:6d 0:3:ff:df:95:84 ip 165: IP (tos 0x10, ttl 62, id 43362, len 151)
172.20.201.135.21 > 10.10.10.228.35886: P
34802:34901(99) ack 16652 win 32120 <nop,nop,timestamp 2548231 356444> (DF)
14:21:30.249665 0:3:ff:df:95:84 0:50:56:40:0:6d ip 66: IP (tos 0x0, ttl 64, id 17789, len 52)
10.10.10.228.35886 > 172.20.201.135.21: . [tcp sum ok] ack 34901 win 20874 <nop,nop,timestamp
356447 2548231> (DF)
```

(A)

(B)

Attacker (10.10.10.228) |-----|-----Gateway-----| Target Host (172.20.201.135)
IDS

A: Ethernet address [0:3:ff:df:95:84] - Connectix

B: Ethernet address [0:50:56:40:0:6d] - VmWare Inc.

The VmWare and Connectix are both the same kind of product used to create virtual machines.

I have used 'A' for Attacker host and 'B' for target host for our discussion.

Alerts were generated by Snort (Windows version) 2.1.0 Build 10 with the latest rules available on Jan. 6 at snort.org.

-X = Dump raw packet data starting at link layer -y = include year in timestamp in the alerts and log files -k = no checksum

The snort rule which has triggered the alert is:

**alert ip any any -> any any (msg:"ATTACK-RESPONSES id check returned root";
content: "uid=0(root)"; classtype:bad-unknown; sid:498; rev:4;)**

The above rule has generated the following alert.

```
[**] [1:498:4] ATTACK-RESPONSES id check returned root [**]  
[Classification: Potentially Bad Traffic] [Priority: 2]  
11/18/03-14:21:28.106376 172.20.201.135:21 -> 10.10.10.228:35886 TCP  
TTL:62 TOS:0x10 ID:43361 IpLen:20 DgmLen:104 DF  
***AP*** Seq: 0x65F61C96 Ack: 0x9BDB6947 Win: 0x7D78 TcpLen: 32 TCP Options (3) => NOP NOP  
TS: 2548019 356238
```

As you see above, this alert could be generated by any IP traffic to any host which has the content "uid=0(root)" in its payload. This looks to be the output of the UNIX command "id".

This signature only looks for content "uid=0(root)". So there is always a chance to be a false positive alert. Any email with "UID=0(root)" in its content could produce the same results. But in this case the attacker has issued the "id" command after a series of SITE EXEC commands, which creates more suspicion of attack.

3) PROBABILITY THE SOURCE ADDRESS WAS SPOOFED

```
windump -r c:\gcia\2003.12.15.12 -e -v host 10.10.10.228 and host  
172.20.201.135 and (tcp port 35886 and tcp port 21)  
14:21:22.084529 0:3:ff:df:95:84 0:50:56:40:0:6d ip 74: IP (tos 0x0, ttl 64, id 17676, len 60)  
10.10.10.228.35886 > 172.20.201.135.21: S [tcp sum ok] 2614831172:2614831172(0) win 5840 <mss  
1460,sackOK,timestamp 355656 0,nop,wscale 0> (DF)  
14:21:22.087897 0:50:56:40:0:6d 0:3:ff:df:95:84 ip 74: IP (tos 0x0, ttl 62, id 43249, len 60)  
172.20.201.135.21 > 10.10.10.228.35886: S [tcp sum ok] 1710593240:1710593240(0) ack 2614831173  
win 32120 <mss 1460,sackOK,timestamp 2547417 355656,nop,wscale 0> (DF)  
14:21:22.088966 0:3:ff:df:95:84 0:50:56:40:0:6d ip 66: IP (tos 0x0, ttl 64, id 17677, len 52)  
10.10.10.228.35886 > 172.20.201.135.21: . [tcp sum ok] ack 1 win 5840 <nop,nop,timestamp 355657  
2547417> (DF)
```

As you can see in above sample traffic, the three-way handshake was completed so a complete connection was established with the target host. The IP address is not spoofed in this case. I have run 'pof' passive fingerprinting tool to find out operating systems involved.

```
Pof -s c:\2003.12.15.12 -o c:\gcia.txt
```

```
<Fri Jan 16 23:51:56 2004> 172.20.201.135:943 - Linux 2.2 (up: 7 hrs)
```

```
<Fri Jan 16 23:54:05 2004> 10.10.10.228:34334 - Linux 2.4/2.6 (up: 0 hrs)  
-> 172.20.201.135:5680 (distance 0, link: Ethernet/modem)
```

Both sides appear to involve Linux operating systems.

4) DESCRIPTION OF ATTACK

This attack is targeted to systems running wuFTPd version 2.6.0 or earlier. WuFTPd is an ftp daemon developed by Washington University.

The Site Exec vulnerability was discovered in 2.6.0 in Oct. 1999 but appears to be existent since 1993. WuFTPd was included with most popular Linux distributions such as Red Hat 6.2, SuSe, FreeBSD, etc.

With "site exec" enabled, a user logged in with anonymous access to an ftp server may execute a restricted subset of quoted commands on the server. Due to insufficient input validation, the attacker can send a specially crafted string (f% f% a%) to override data on the stack. Once exploited successfully, the attacker can execute commands with root access. With root access the attacker can view or alter any system files to gain more access to the system. Several exploit scripts were developed to take advantage of this vulnerability, such as wuftp2600.c, or bobek.c, etc.

CERT Advisory (CA-2000-13) was also issued for this vulnerability.

5) ATTACK MECHANISM

I have used many other tools such as ethereal, tcpdump, and snort to analyze traffic, but I have only included specific sample traffic to save space.

```
C:\Snort\bin>snort -r c:\gcia\2003.12.15.12 -l c:\snort\log -y -d -e -v
```

Options:

-r: Read and process TCPDUMP file

-l: Log to directory

-y: Include year in Timestamp

-d: Dump Application Layer

-e: Dump Layer 2 information

-v: Be verbose

```
=====  
11/18/03-14:21:22.170999 0:3:FF:DF:95:84 -> 0:50:56:40:0:6D type:0x800 len:0x4B  
10.10.10.228:35886 -> 172.20.201.135:21 TCP TTL:64 TOS:0x0 ID:17679 IpLen:20 DgmLen:61 DF  
***AP*** Seq: 0x9BDB2845 Ack: 0x65F59542 Win: 0x16D0 TcpLen: 32 TCP Options (3) => NOP NOP  
TS: 355664 2547425  
55 53 45 52 20 66 74 70 0A          USER ftp.
```

```
=====  
11/18/03-14:21:22.183677 0:50:56:40:0:6D -> 0:3:FF:DF:95:84 type:0x800  
len:0x42  
172.20.201.135:21 -> 10.10.10.228:35886 TCP TTL:62 TOS:0x10 ID:43257 IpLen:20 DgmLen:52 DF  
***A*** Seq: 0x65F59542 Ack: 0x9BDB284E Win: 0x7D78 TcpLen: 32 TCP Options (3) => NOP NOP  
TS: 2547426 355664
```

```
=====
```

```
11/18/03-14:21:22.184793 0:50:56:40:0:6D -> 0:3:FF:DF:95:84 type:0x800
len:0x86
172.20.201.135:21 -> 10.10.10.228:35886 TCP TTL:62 TOS:0x10 ID:43258 IpLen:20 DgmLen:120 DF
***AP*** Seq: 0x65F59542 Ack: 0x9BDB284E Win: 0x7D78 TcpLen: 32 TCP Options (3) => NOP NOP
TS: 2547426 355664
33 33 31 20 47 75 65 73 74 20 6C 6F 67 69 6E 20 331 Guest login
6F 6B 2C 20 73 65 6E 64 20 79 6F 75 72 20      ok. send your
```

```
11/18/03-14:21:22.187651 0:3:FF:DF:95:84 -> 0:50:56:40:0:6D type:0x800 len:0x50
10.10.10.228:35886 -> 172.20.201.135:21 TCP TTL:64 TOS:0x0 ID:17680 IpLen:20 DgmLen:66 DF
***AP*** Seq: 0x9BDB284E Ack: 0x65F59586 Win: 0x16D0 TcpLen: 32 TCP Options (3) => NOP NOP
TS: 355665 2547426
50 41 53 53 20 6D 6E 7A 69 6C 6C 61 40 0A PASS mozilla@.
```

```
11/18/03-14:21:22.198227 0:50:56:40:0:6D -> 0:3:FF:DF:95:84 type:0x800
len:0x72
172.20.201.135:21 -> 10.10.10.228:35886 TCP TTL:62 TOS:0x10 ID:43259 IpLen:20 DgmLen:100 DF
***AP*** Seq: 0x65F59586 Ack: 0x9BDB285C Win: 0x7D78 TcpLen: 32 TCP Options (3) => NOP NOP
TS: 2547428 355665
32 33 30 20 47 75 65 73 74 20 6C 6F 67 69 6E 20 230 Guest login
6F 6B 2C 20 61 63 63 65 73 73 20 72 65 73      ok, access res
```

The attacker was successfully logged in to the target FTP server with a guest login.

```

11/18/03-14:21:22.209882 0:3:FF:DF:95:84 -> 0:50:56:40:0:6D type:0x800 len:0x5A
10.10.10.228:35886 -> 172.20.201.135:21 TCP TTL:64 TOS:0x0 ID:17681 IpLen:20 DgmLen:76 DF
***AP*** Seq: 0x9BDB285C Ack: 0x65F595B6 Win: 0x16D0 TcpLen: 32 TCP Options (3) => NOP NOP
TS: 355668 2547428
53 49 54 45 20 45 58 45 43 20 25 30 32 30 64 7C SITE EXEC %020d|
25 2E 66 25 2E 66 7C 0A %f.%f|.

```

```
11/18/03-14:21:22.215740 0:50:56:40:0:6D -> 0:3:FF:DF:95:84 type:0x800
len:0x61
172.20.201.135:21 -> 10.10.10.228:35886 TCP TTL:62 TOS:0x10 ID:43260 IpLen:20 DgmLen:83 DF
***AP*** Seq: 0x65F595B6 Ack: 0x9BDB2874 Win: 0x7D78 TcpLen: 32 TCP Options (3) => NOP NOP
TS: 2547430 355668
32 30 30 2D 30 30 30 30 30 30 30 30 30 30 30 200-00000000000000
30 30 30 30 30 30 34 39 7C 30 2D 32 7C 0D 0000004910-2|.
```

After successfully logging in as "guest," The attacker was issuing SITE EXEC command, trying to exploit a vulnerability in FTP server (wuftp version 2.6.0 or earlier). The above traffic has generated snort alerts "FTP site exec" (ALERT 1).

```

==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+
11/18/03-14:21:28.091131 0:3:FF:DF:95:84 -> 0:50:56:40:0:6D type:0x800
len:0x46
10.10.10.228:35886 -> 172.20.201.135:21 TCP TTL:64 TOS:0x0 ID:17786 IpLen:20 DgmLen:56 DF
***AP*** Seq: 0x9BDB6943 Ack: 0x65F61C96 Win: 0x518A TcpLen: 32 TCP Options (3) => NOP NOP
TS: 356238 2547819
69 64 3B 0A id;.

```

```

==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+
11/18/03-14:21:28.106376 0:50:56:40:0:6D -> 0:3:FF:DF:95:84 type:0x800
len:0x76
172.20.201.135:21 -> 10.10.10.228:35886 TCP TTL:62 TOS:0x10 ID:43361 IpLen:20 DgmLen:104 DF
***AP*** Seq: 0x65F61C96 Ack: 0x9BDB6947 Win: 0x7D78 TcpLen: 32 TCP Options (3) => NOP NOP
TS: 2548019 356238
75 69 64 3D 30 28 72 6F 6F 74 29 20 67 69 64 3D uid=0(root) gid=
30 28 72 6F 6F 74 29 20 65 67 69 64 3D 35 0(root) egid=5

```

```

==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+
11/18/03-14:21:28.107735 0:3:FF:DF:95:84 -> 0:50:56:40:0:6D type:0x800
len:0x42
10.10.10.228:35886 -> 172.20.201.135:21 TCP TTL:64 TOS:0x0 ID:17787 IpLen:20 DgmLen:52 DF
***A*** Seq: 0x9BDB6947 Ack: 0x65F61CCA Win: 0x518A TcpLen: 32 TCP Options (3) => NOP NOP
TS: 356239 2548019

```

```

==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+
11/18/03-14:21:30.222374 0:3:FF:DF:95:84 -> 0:50:56:40:0:6D type:0x800 len:0x4B
10.10.10.228:35886 -> 172.20.201.135:21 TCP TTL:64 TOS:0x0 ID:17788 IpLen:20 DgmLen:61 DF
***AP*** Seq: 0x9BDB6947 Ack: 0x65F61CCA Win: 0x518A TcpLen: 32 TCP Options (3) => NOP NOP
TS: 356444 2548019
75 6E 61 6D 65 20 2D 61 0A uname -a.

```

```

==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+
11/18/03-14:21:30.237991 0:50:56:40:0:6D -> 0:3:FF:DF:95:84 type:0x800
len:0xA5
172.20.201.135:21 -> 10.10.10.228:35886 TCP TTL:62 TOS:0x10 ID:43362 IpLen:20 DgmLen:151 DF
***AP*** Seq: 0x65F61CCA Ack: 0x9BDB6950 Win: 0x7D78 TcpLen: 32 TCP Options (3) => NOP NOP
TS: 2548231 356444 4C 69 6E 75 78 20 31 37 32 2D
32 30 2D 32 30 31 Linux 172-20-201
2D 31 33 35 2E 4D 53 59 2D 50 4F 50 2E 49 -135.MSY-POP.I

```

```

==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+
11/18/03-14:21:30.249665 0:3:FF:DF:95:84 -> 0:50:56:40:0:6D type:0x800
len:0x42
10.10.10.228:35886 -> 172.20.201.135:21 TCP TTL:64 TOS:0x0 ID:17789 IpLen:20 DgmLen:52 DF
***A*** Seq: 0x9BDB6950 Ack: 0x65F61D2D Win: 0x518A TcpLen: 32 TCP Options (3) => NOP NOP
TS: 356447 2548231

```

```

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

```

Next, the attacker issued the `id` UNIX command. The target has replied with a `uid=0(root)`. Looking at the data in packets, `uid=0(root)` has set a snort trigger for "ATTACK-RESPONSES id check returned root". The `uid=0` indicates super user rights on the UNIX system, which verifies that the attacker has root access on the target machine. Then the attacker issued the additional command "`uname -a`" to find out more information about the target host. At this point, the attacker was able to successfully

exploit SITE EXEC vulnerability on target ftp server and able to gain root access to the target system.

6) CORRELATIONS

CVE,

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0573>

Security focus,

<http://www.securityfocus.com/bid/1387> and
<http://www.securityfocus.com/bid/2241/exploit/>

ISS,

<http://xforce.iss.net/xforce/xfdb/4773>

This kind of attack was referenced in a GCIA practical by Chris Compton.

<http://cert.uni-stuttgart.de/archive/intrusions/2004/01/msg00020.html>

7) EVIDENCE OF ACTIVE TARGETING

C:\>windump -r c:\gcia\2003.12.15.12 -v host 10.10.10.228 and host
172.20.201.135

14:20:47.664964 IP (tos 0x0, ttl 39, id 59086, len 60) 10.10.10.228.43995 > 172.20.201.135.21: SE [tcp sum ok] 2174667858:2174667858(0) win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
14:20:47.665323 IP (tos 0x0, ttl 39, id 32587, len 60) 10.10.10.228.43996 > 172.20.201.135.21: . [tcp sum ok] win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
14:20:47.665654 IP (tos 0x0, ttl 39, id 13155, len 60) 10.10.10.228.43997 > 172.20.201.135.21: SFP [tcp sum ok] 2174667858:2174667858(0) win 4096 urg 0 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
14:20:47.665850 IP (tos 0x0, ttl 39, id 6315, len 60) 10.10.10.228.43998 > 172.20.201.135.21: . [tcp sum ok] ack 0 win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
14:20:47.666083 IP (tos 0x0, ttl 39, id 27019, len 60) 10.10.10.228.43999 > 172.20.201.135.1: S [tcp sum ok] 2174667858:2174667858(0) win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
14:20:47.666274 IP (tos 0x0, ttl 39, id 30187, len 60) 10.10.10.228.44000 > 172.20.201.135.1: . [tcp sum ok] ack 0 win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
14:20:47.666726 IP (tos 0x0, ttl 39, id 30287, len 60) 10.10.10.228.44001 > 172.20.201.135.1: FP [tcp sum ok] 2174667858:2174667858(0) win 4096 urg 0 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
14:20:47.667355 IP (tos 0x0, ttl 61, id 10393, len 328) 10.10.10.228.43988 > 172.20.201.135.1: udp 300
14:20:47.673122 IP (tos 0x0, ttl 62, id 43148, len 60) 172.20.201.135.21 > 10.10.10.228.43995: S [tcp sum ok] 1675250557:1675250557(0) ack 2174667859 win 32595 <mss 265,nop,nop,timestamp 2543976 1061109567,nop,wscale 0> (DF)
14:20:47.674000 IP (tos 0x0, ttl 64, id 0, len 40) 10.10.10.228.43995 > 172.20.201.135.21: R [tcp sum ok] 2174667859:2174667859(0) win 0 (DF)
14:20:47.677190 IP (tos 0x0, ttl 62, id 43149, len 60) 172.20.201.135.21 > 10.10.10.228.43997: S [tcp sum ok] 1672795107:1672795107(0) ack 2174667859 win 32595 <mss 265,
.....

As you can see in the above traffic, the attacking host was actively targeting many hosts. Also notice that the attacker has done a comprehensive scan on the target system.

8) SEVERITY

To calculate the severity of these alerts I have followed the guidelines available at: http://www.giac.org/GCIA_assignment.php.

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

Criticality: measure of how critical the target system is.

FTP is involved in file transfer and users may be using for transferring critical files with customers. The guest access to the FTP Sever should not be allowed anyway. This exploit gives attackers Root access so attackers may use this system to exploit other systems.

Criticality=4

Lethality: how severe the damage to targeted system would be if attack occurred.

The attacker has root super user access to the system. At this point attacker can change any system files to gain more access or can download a backdoor to the system.

Lethality=5

System Countermeasures: measure of the strength of defensive mechanisms in place on host itself.

This is a vulnerability that exists in old versions of wuFTP 2.6.0 or earlier which comes with older Linux distributions.

System Countermeasures=1

Network Counter Measures: measure of defensive mechanisms in place on network such as firewall.

The guest access to ftp server should not be allowed to the ftp server. I do not see any evidence of a firewall present.

Network Counter Measures=1

Severity= (4+5)-(1+1) = 7

This alert needs immediate attention. Need to block any traffic from this host at the firewall until further investigation.

9) DEFENSIVE RECOMMENDATION

Disable FTP service or upgrade ftpd to newer version.

FTP transfers user credentials in clear text anyway, so move to a different solution such as SFTP or SSL/TLS supported ftp servers. The host hardening and hot fix is really important in the event, if you really have to run FTP on server.

10) Multiple choice question

Q: Which internet worm were targets common vulnerabilities in WuFTP on Linux systems?

- a) Adore Worm
- b) Poison box
- c) Code red
- d) Nachi worm

Ans: a

DETECT: 2 PORT 0 TRAFFICS

(Posted on: 1/13/2004)

1) SOURCE OF TRACE:

The Raw tcpdump logs were obtained from <http://www.incident.org/logs/raw> website. The archive file 2003.12.15.tgz contained many days worth of logs files, but 2003.12.15.7 is used for this analysis. The IP address in the log files has been change to hide the real identity.

I have used windump, to find out information about the unknown network topology.

C:\>windump -r c:\gcia\2003.12.15.7 -e -v (host 10.10.10.141 and (dst port 0)) or (host 172.20.11.2 and (src port 0))

```
14:09:20.939976 0:d0:59:c6:5e:14 0:50:56:40:0:6d ip 74: IP (tos 0x0, ttl 64, id 58896, len 60)
10.10.10.141.34054 > 172.20.11.2.0: S [tcp sum ok] 3321649537:3321649537(0) win 5840 <mss
1460,sackOK,timestamp 49041 0,nop,wscale 0> (DF)
14:09:20.984175 0:50:56:40:0:6d 0:d0:59:c6:5e:14 ip 60: IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 >
10.10.10.141.34054: R [tcp sum ok] 0:0(0) ack 3321649538 win 0
14:09:22.810167 0:d0:59:c6:5e:14 0:50:56:40:0:6d ip 60: IP (tos 0x0, ttl 255, id 47626, len 40)
10.10.10.141.17012 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
14:09:22.820291 0:50:56:40:0:6d 0:d0:59:c6:5e:14 ip 60: IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 >
10.10.10.141.17012: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)
14:09:22.823380 0:d0:59:c6:5e:14 0:50:56:40:0:6d ip 60: IP (tos 0x0, ttl 255, id 47626, len 40)
10.10.10.141.11113 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
14:09:22.829898 0:50:56:40:0:6d 0:d0:59:c6:5e:14 ip 60: IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 >
10.10.10.141.11113: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)
14:09:22.830291 0:d0:59:c6:5e:14 0:50:56:40:0:6d ip 60: IP (tos 0x0, ttl 255, id 47626, len 40)
10.10.10.141.25218 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
14:09:22.840616 0:50:56:40:0:6d 0:d0:59:c6:5e:14 ip 60: IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 >
10.10.10.141.25218: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)
14:09:22.844382 0:d0:59:c6:5e:14 0:50:56:40:0:6d ip 60: IP (tos 0x0, ttl 255, id 47626, len 40)
10.10.10.141.45474 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
14:09:22.888300 0:50:56:40:0:6d 0:d0:59:c6:5e:14 ip 60: IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 >
10.10.10.141.45474: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)
14:09:22.888869 0:d0:59:c6:5e:14 0:50:56:40:0:6d ip 60: IP (tos 0x0, ttl 255, id 47626, len 40)
10.10.10.141.63176 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
```



```

14:09:22.898549 0:50:56:40:0:6d 0:d0:59:c6:5e:14 ip 60: IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 >
10.10.10.141.63176: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)
14:09:22.899069 0:d0:59:c6:5e:14 0:50:56:40:0:6d ip 60: IP (tos 0x0, ttl 255, id 47626, len 40)
10.10.10.141.62917 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
14:09:22.939627 0:50:56:40:0:6d 0:d0:59:c6:5e:14 ip 60: IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 >
10.10.10.141.62917: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)

```

0:d0:59:c6:5e:14 : Ambit micro system
0:50:56:40:0:6d : VmWare inc.

I have used the website <http://standards.ieee.org/regauth/oui/index.shtml> to find the network card manufacturer for this host.

Speculated Network Topology:

```

(A)                                     (B)
Attacker (10.10.10.141) |-----|-----Gateway (10.10.10.1)-----| Target Host
(172.20.11.2)           IDS

```

NOTE:

I have used 'A' for Attacker host and 'B' for target host for our discussion.

2) DETECT WAS GENERATED BY:

Alerts were generated by snort (windows version) 2.1.0 Build 10 with latest rule available on Jan. 6 at snort.org.

```

C:\snort\bin>snort -r c:\gcia\2003.12.15.7 -c c:\snort\etc\snort.conf -A full -l c:\snort\log -
X -y -k none

```

Options Details:

```

-r = Read tcpdump file
-c = rule file specified
-A = set Alert mode to full
-X = Dump raw packet data starting at link layer
-y = include year in timestamp in the alerts and log files
-k = no checksum

```

Here is some example of triggers on traffic:

```

-----
[**] [1:524:7] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/18/03-14:09:20.939976 10.10.10.141:34054 -> 172.20.11.2:0
TCP TTL:64 TOS:0x0 ID:58896 IpLen:20 DgmLen:60 DF
*****S* Seq: 0xC5FC5981 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 49041 0 NOP WS: 0

```

```

[**] [1:524:7] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/18/03-14:09:20.984175 172.20.11.2:0 -> 10.10.10.141:34054

```

TCP TTL:62 TOS:0x0 ID:0 IpLen:20 DgmLen:40 DF
***A*R** Seq: 0x0 Ack: 0xC5FC5982 Win: 0x0 TcpLen: 20

[**] [1:524:7] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/18/03-14:09:22.810167 10.10.10.141:17012 -> 172.20.11.2:0
TCP TTL:255 TOS:0x0 ID:47626 IpLen:20 DgmLen:40
*****S* Seq: 0xF1C Ack: 0x0 Win: 0x200 TcpLen: 20

[**] [1:524:7] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/18/03-14:09:22.820291 172.20.11.2:0 -> 10.10.10.141:17012
TCP TTL:62 TOS:0x0 ID:0 IpLen:20 DgmLen:40 DF
***A*R** Seq: 0x0 Ack: 0xF1D Win: 0x0 TcpLen: 20

The Snort rules which triggered this alert are as follow:

alert tcp \$EXTERNAL_NET any <> \$HOME_NET 0 (msg:"BAD-TRAFFIC tcp port 0 traffic"; stateless; classtype:misc-activity; sid:524; rev:7;)

Alerts were triggered based upon destination port 0. In IP Communication port 0 is reserved so it should not be used for any normal flow of traffic. All operation systems react differently for traffic received on port 0. It is dependent upon the operation system. Some operating systems respond with RST packets and some do not respond to port 0 traffic. Attacker can utilize this behavior for fingerprinting the OS. My first windump data shows that the attacker has made 7 connection attempts to the target host within 2 seconds. I can also see response traffic (RST, ACK) from the target host which means the target host appears to be live.

3) PROBABILITY THE SOURCE WAS SPOOFED

I have used windump to analyze Layer 2 frames.

c:\windump -r c:\gcia\2003.12.15.7 -e -v host 10.10.10.141

14:08:43.858956 0:d0:59:c6:5e:14 0:50:56:40:0:6d ip 74: IP (tos 0x0, ttl 64, id 3036, len 60)
10.10.10.141.32794 > 172.20.11.2.1: S [tcp sum ok] 3291658953:3291658953(0) win 5840 <mss
1460,sackOK,timestamp 45333 0,nop,wscale 0> (DF)
14:08:43.859010 0:d0:59:c6:5e:14 0:50:56:40:0:6d ip 74: IP (tos 0x0, ttl 64, id 7702, len 60)
10.10.10.141.32795 > 172.20.11.2.2: S [tcp sum ok] 3287199491:3287199491(0) win 5840 <mss
1460,sackOK,timestamp 45333 0,nop,wscale 0> (DF)
14:08:43.859013 0:d0:59:c6:5e:14 0:50:56:40:0:6d ip 74: IP (tos 0x0, ttl 64, id 51986, len 60)
10.10.10.141.32796 > 172.20.11.2.3: S [tcp sum ok] 3281680903:3281680903(0) win 5840 <mss
1460,sackOK,timestamp 45333 0,nop,wscale 0> (DF)
14:08:43.859061 0:d0:59:c6:5e:14 0:50:56:40:0:6d ip 74: IP (tos 0x0, ttl 64, id 49220, len 60)
10.10.10.141.32797 > 172.20.11.2.5: S [tcp sum ok] 3287576913:3287576913(0) win 5840 <mss
1460,sackOK,timestamp 45333 0,nop,wscale 0> (DF)
14:08:43.859096 0:d0:59:c6:5e:14 0:50:56:40:0:6d ip 74: IP (tos 0x0, ttl 64, id 9261, len 60)
10.10.10.141.32798 > 172.20.11.2.7: S [tcp sum ok] 3284374840:3284374840(0) win 5840 <mss
1460,sackOK,timestamp 45333 0,nop,wscale 0> (DF)

```
....  
14:09:20.984175 0:50:56:40:0:6d 0:d0:59:c6:5e:14 ip 60: IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 >  
10.10.10.141.34054: R [tcp sum ok] 0:0(0) ack 3321649538 win 0  
14:09:22.810167 0:d0:59:c6:5e:14 0:50:56:40:0:6d ip 60: IP (tos 0x0, ttl 255, id 47626, len 40)  
10.10.10.141.17012 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
```

The attacker has initiated many different connections to other hosts on the network. The Ethernet address in all connections was the same IP address. You can also mark ttl values (64), which is the same for all connections. That means the IP address does not look to be spoofed. Also if the attacker wants to fingerprint the target host then he has to wait for any response traffic unless the attacker and the target are on the same network. If the attacker is on the same network then he can sniff traffic for responses.

4) ATTACK DESCRIPTION

The port 0 is reserved for special use as defined in RFC 1700. Any traffic should not flow to port 0. If you see any traffic from or destined to port 0 then it is not a normal activity. Most operating systems behave differently to port 0 traffic so the system could be fingerprinted.

Here is an example I have found at

<http://www.securiteam.com/securityreviews/5XP0Q2AAKS.html> , Article by Ste Jones.

Fingerprint OpenBSD 3.2/3.3

P1(Resp=Y%Flags=AR)

P2(Resp=Y%Flags=AR)

P3(Resp=N)

P4(Resp=Y%Flags=AR)

P5(Resp=N)

P6(Resp=N)

P7(Resp=Y)

Notice that OpenBSD has a feature/bug whereby it does not allow incoming connections from source port 0 (test P3)

Fingerprint Linux

P1(Resp=Y%Flags=AR)

P2(Resp=Y%Flags=AR)

P3(Resp=Y%Flags=AS)

P4(Resp=Y%Flags=AR)

P5(Resp=Y)

P6(Resp=Y)

P7(Resp=Y)

Unfortunately, both MS Windows 2000 and Linux have the same port 0 fingerprint, replying to all 7 tests.

I have used windump for further analysis:

```
C:\>windump -r c:\gcia\2003.12.15.7 -v (host 10.10.10.141 and (dst port 0)) or (host 172.20.11.2 and (src port 0))
```

```
14:09:20.939976 IP (tos 0x0, ttl 64, id 58896, len 60) 10.10.10.141.34054 > 172.20.11.2.0: S [tcp sum ok] 3321649537:3321649537(0) win 5840 <mss 1460,sackOK,timestamp 49041 0,nop,wscale 0> (DF)
14:09:20.984175 IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 > 10.10.10.141.34054: R [tcp sum ok] 0:0(0) ack 3321649538 win 0 (DF)
14:09:22.810167 IP (tos 0x0, ttl 255, id 47626, len 40) 10.10.10.141.17012 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
14:09:22.820291 IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 > 10.10.10.141.17012: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)
14:09:22.823380 IP (tos 0x0, ttl 255, id 47626, len 40) 10.10.10.141.11113 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
14:09:22.829898 IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 > 10.10.10.141.11113: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)
14:09:22.830291 IP (tos 0x0, ttl 255, id 47626, len 40) 10.10.10.141.25218 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
14:09:22.840616 IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 > 10.10.10.141.25218: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)
14:09:22.844382 IP (tos 0x0, ttl 255, id 47626, len 40) 10.10.10.141.45474 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
14:09:22.888300 IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 > 10.10.10.141.45474: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)
14:09:22.888869 IP (tos 0x0, ttl 255, id 47626, len 40) 10.10.10.141.63176 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
14:09:22.898549 IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 > 10.10.10.141.63176: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)
14:09:22.899069 IP (tos 0x0, ttl 255, id 47626, len 40) 10.10.10.141.62917 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
14:09:22.939627 IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 > 10.10.10.141.62917: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)
```

The attacker has sent 7 packets destined to port 0. Please mark sequence number and IP id values in the above traffic excluding the first connection. The same values for sequence number and id make me think this traffic is crafted with some tool such as Hping or gobbler. First thought might be that this is the same connection attempt because it has the same sequence numbers, but it is not. The Source port is changed for all other connections. I can also see that the target system has responded with RST, ACK packets.

Here is detail packet information.

```
C:\snort\bin>snort -r c:\gcia\2003.12.15.7 -c c:\snort\etc\snort.conf -A full -l c:\snort\log -X -y -k none
```

5) ATTACK MECHANISM

The hping application is available at website <http://www.hping.org>. The Hping 2.0 Release Candidate 2 is available now. I believe the following command will reproduce same alert as above.

Options used:

For more options please refer to man page at <http://www.hping.org>. The Hping by default set destination port to 0 unless you specified with -p <port> option.

6) CORRELATIONS

I have found the same detection at website <http://cert.uni-stuttgart.de/archive/intrusions/2003/09/msg00030.html>. This identifies Hping as a tool used for this kind of alert. The author also discusses use of Perl and RawIP module to craft packets. I agree with him too. It is also possible that the attacker might have utilized some kind of Perl script with the RawIP module to craft this kind of traffic. Patenaude Patrick has also analyzed the same alert at <http://cert.uni-stuttgart.de/archive/intrusions/2003/04/msg00074.html>. In his article he is also leaning towards automated tools such as Hping or Nmap. Nmap also has the capability to scan port 0 after version 3.2.

Edwin Fung has also analyzed the same alert at http://www.giac.org/practical/GCIA/Ewen_Fung_GCIA.pdf in his assignment. He also concluded Hping2 as a tool used to craft packets.

I agree with all of them for this alert. The Hping2 is probably being used for these alerts. These days many tools are available for crafting packets but Hping is more popular and easy to use. I have also found that Checkpoint firewall has vulnerability for UDP port 0 traffic as discussed in <http://www.securiteam.com/exploits/2UUQCRFS00.html>.

7) EVIDENCE OF ACTIVE TARGETING

```
14:09:22.810167 IP (tos 0x0, ttl 255, id 47626, len 40) 10.10.10.141.17012 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
14:09:22.820291 IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 > 10.10.10.141.17012: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)
14:09:22.823380 IP (tos 0x0, ttl 255, id 47626, len 40) 10.10.10.141.11113 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
14:09:22.829898 IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 > 10.10.10.141.11113: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)
14:09:22.830291 IP (tos 0x0, ttl 255, id 47626, len 40) 10.10.10.141.25218 > 172.20.11.2.0: S [tcp sum ok] 3868:3868(0) win 512
14:09:22.840616 IP (tos 0x0, ttl 62, id 0, len 40) 172.20.11.2.0 > 10.10.10.141.25218: R [tcp sum ok] 0:0(0) ack 3869 win 0 (DF)
```

After look at above traffic it is certain that this is an active fingerprinting attempt. In a traffic the ttl values and sequence number was staying same and also target host was reacting for those connections with RST, ACK response.

8) SEVERITY

To calculate severity of these alerts I have followed the following guidelines which are available at http://www.giac.org/GCIA_assignment.php.

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

Criticality: measure of how critical the target system is.

I do not have any more information on the target host about how important this is. Additionally port 0 traffic should be filtered for any important host anyway. But in this case the host is responding with RST packets which means the filtering device did not exist or rules were not set up to filter port 0 traffic.

Criticality = 2

Lethality: how severe the damage to targeted system would be if attack occurred.

This looks to be that the attacker is doing reconnaissance at this point. It is safe for me to assume at this point 2, but in reality this type of scan is common so we can give it a 1.

System Countermeasures: measure of the strength of defensive mechanisms in place on host itself.

The target is replying with RST, ACK packets. This is normal behavior for most systems. There is no payload in the data so I can not tell any application was listening on port 0.

System countermeasures= 3

Network Counter Measures: measure of defensive mechanisms in place on network such as firewall.

As I stated before, there does not seem to be any filtering device in existence on this network.

Network countermeasures= 2

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

This rating does not require any further investigation at this stage.

9) DEFENSE RECOMMENDATIONS

Most firewalls will drop port 0 traffic with default drop rules. But the access list on the router will need to be set up to block any traffic which comes to port 0.

10) MULTIPLE CHOICE QUESTION

```
[root@localhost root]# hping -a 172.16.20.1 -S 10.10.8.142
HPING 10.10.8.142 (eth0 10.10.8.142): S set, 40 headers + 0 data bytes
```

```
--- 10.10.8.142 hping statistic ---
19 packets remitted, 0 packets received, 100% packet loss
Round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Why do we not see any reply from target host which is on the same network when no firewall exists between them?

- A) Command is wrong.
- B) With -a option we told Hping to spoof the source address
- C) Because the host is on the same broadcast domain

Ans: B

QUESTIONS & ANSWERS:

From Joe Bowling (joebowling@comcast.net) Ask multiple questions in several sections:

Q1) can you please describe how the differ OS respond to such packets?

ANS: Most system react with reset except some version of SUN OS (? I am not sure which version) that do not respond.

Q2) I am not sure how the above (spoofing section)statement confirms that the Packets are/aren't spoofed.....you may want to mention the connection type nature of TCP. Also with tools such as Hping it could be possible to do partial OS fingerprinting off of the silent 3rd party (tangent I know but FYI)

ANS: You are also right. It is hard to guess IP address was spoofed or not. I can see other three-way handshake traffic that's why I am gassing IP address not spoofed. But it also possible that might be a 3rd host involved in this and craft packets with attackers MAC and IP Address.

Q3) Have you downloaded hping and tried to recreate the results? The seq numbers in the traffic mentioned the sections prior show changing seq numbers?

ANS: The packet which has trigger rules had a same sequence number and IP ID number except for first packet. After I analyze pattern in detail I realize that these packets may not be crafted with Hping. The Hping always increment sequence number by one and in this case sequence number was constant. This smell like libnet script. The attacker might have used custom libnet script or automate nemesis with script. I have download nemesis and I was able to recreate patterns.

```
nemesis tcp -H 00:d0:59:c6:5e:14 -M 00:50:56:40:00:6d -t 0x0 -T 255 -I 47626 -S 10.10.10.141 -D 172.20.11.2 -x 17012 -y 0 -s 3868 -w 512 -fS -a 0
```

Options are as follow:

-H : Source MAC Address

-M : Destination MAC Address

-t : TOS

-T : ttl

-I : IP ID

-S : Source IP

-D : Destination IP

-x : Source Port

-y : Destination Port

-s : seq. number

-w : Window size

-fs: SYN packet

-a : ack number

I was able to make batch file to run this command and recreate same traffic. It also easy to automate nemesis with vb or Perl script.

Q4) So you think its ok for this guy to scan (or as you believe fingerprint) hosts in the internal network??? I suggest we get a firewall to block such traffic in to begin with and would highly consider blocking the offending IP temporarily...Dshield show

Anything on offenders IP?

ANS: In this alerts appears to be a all local network that setup on Vmware because both are private IP address 10.10.x and 172.20.X . In this case I assume that this is local network and administrator may be scanning for security testing. In reality also I might be block that IP address until further analyze.

From Johnny Wong, Ask Multiple questions in several sections:

Q5) If you look at the rule, it does not trigger just on destination port alone.

The "<>" means 2-way, so a source port of 0 might also trigger.

ANS: The snort rules should be triggered for either source or destination port 0 traffic.

The port 0 traffic in either direction is not normal.

DETECT: 3 UDP Scan by ISS

1) SOURCE OF TRACE

The Raw tcpdump logs were obtained from <http://www.incident.org/logs/raw> website.

The archive file 2003.12.15.tgz contained several days' worth of log files, but 2003.12.15.8 is used for this analysis. The IP addresses shown in log file have been changed from real to hide identity. By using windump, I have learned information at the IP level about the unknown network.

C:\>windump -r c:\2003.12.15.8 -v -e host 10.10.10.165 and host 172.20.201.198

```
14:10:14.171828 0:3:47:8c:89:c2 0:50:56:40:0:6d ip 61: IP (tos 0x0, ttl 128,
id 44697, len 47) 10.10.10.165.2022 > 172.20.201.198.236: [udp sum ok] udp 19
14:10:14.171898 0:3:47:8c:89:c2 0:50:56:40:0:6d ip 61: IP (tos 0x0, ttl 128,
id 44698, len 47) 10.10.10.165.2022 > 172.20.201.198.237: [udp sum ok] udp 19
14:10:14.171942 0:3:47:8c:89:c2 0:50:56:40:0:6d ip 61: IP (tos 0x0, ttl 128,
id 44699, len 47) 10.10.10.165.2022 > 172.20.201.198.238: [udp sum ok] udp 19
14:10:14.171982 0:3:47:8c:89:c2 0:50:56:40:0:6d ip 61: IP (tos 0x0, ttl 128,
id 44700, len 47) 10.10.10.165.2022 > 172.20.201.198.239: [udp sum ok] udp 19
14:10:14.172024 0:3:47:8c:89:c2 0:50:56:40:0:6d ip 61: IP (tos 0x0, ttl 128,
id 44701, len 47) 10.10.10.165.2022 > 172.20.201.198.240: [udp sum ok] udp 19
.....
14:10:14.196294 0:50:56:40:0:6d 0:3:47:8c:89:c2 ip 89: IP (tos 0xc0, ttl 253,
id 42550, len 75) 172.20.201.198 > 10.10.10.165: icmp 55: 172.20.201.198 udp
port 236 unreachable for IP (tos 0x0, ttl 126, id 44697, len 47)
10.10.10.165.2022 > 172.20.201.198.236: [udp sum ok] udp 19
14:10:14.198843 0:50:56:40:0:6d 0:3:47:8c:89:c2 ip 89: IP (tos 0xc0, ttl 253,
id 42551, len 75) 172.20.201.198 > 10.10.10.165: icmp 55: 172.20.201.198 udp
port 237 unreachable for IP (tos 0x0, ttl 126, id 44698, len 47)
10.10.10.165.2022 > 172.20.201.198.237: [udp sum ok] udp 19
14:10:14.200707 0:50:56:40:0:6d 0:3:47:8c:89:c2 ip 89: IP (tos 0xc0, ttl 253,
id 42552, len 75) 172.20.201.198 > 10.10.10.165: icmp 55: 172.20.201.198 udp
port 238 unreachable for IP (tos 0x0, ttl 126, id 44699, len 47)
10.10.10.165.2022 > 172.20.201.198.238: [udp sum ok] udp 19
14:10:14.202078 0:50:56:40:0:6d 0:3:47:8c:89:c2 ip 89: IP (tos 0xc0, ttl 253,
id 42553, len 75) 172.20.201.198 > 10.10.10.165: icmp 55: 172.20.201.198 udp
port 239 unreachable for IP (tos 0x0, ttl 126, id 44700, len 47)
10.10.10.165.2022 > 172.20.201.198.239: [udp sum ok] udp 19
```

```

14:10:14.202945 0:50:56:40:0:6d 0:3:47:8c:89:c2 ip 89: IP (tos 0xc0, ttl 253,
id 42554, len 75) 172.20.201.198 > 10.10.10.165: icmp 55: 172.20.201.198 udp
port 240 unreachable for IP (tos 0x0, ttl 126, id 44701, len 47)
10.10.10.165.2022 > 172.20.201.198.240: [udp sum ok] udp 19
.....
14:10:19.173617 0:3:47:8c:89:c2 0:50:56:40:0:6d ip 61: IP (tos 0x0, ttl 128,
id 44814, len 47) 10.10.10.165.2022 > 172.20.201.198.2050: [udp sum ok] udp
19 14:10:19.173680 0:3:47:8c:89:c2 0:50:56:40:0:6d ip 61: IP (tos 0x0, ttl
128, id 44815, len 47) 10.10.10.165.2022 > 172.20.201.198.2051: [udp sum ok]
udp 19
14:10:19.173738 0:3:47:8c:89:c2 0:50:56:40:0:6d ip 61: IP (tos 0x0, ttl 128,
id 44816, len 47) 10.10.10.165.2022 > 172.20.201.198.2052: [udp sum ok] udp
19
14:10:19.173779 0:3:47:8c:89:c2 0:50:56:40:0:6d ip 61: IP (tos 0x0, ttl 128,
id 44817, len 47) 10.10.10.165.2022 > 172.20.201.198.2053: [udp sum ok] udp
19
.....
14:10:19.201202 0:50:56:40:0:6d 0:3:47:8c:89:c2 ip 89: IP (tos 0xc0, ttl 253,
id 42601, len 75) 172.20.201.198 > 10.10.10.165: icmp 55: 172.20.201.198 udp
port 2050 unreachable for IP (tos 0x0, ttl 126, id 44814, len 47)
10.10.10.165.2022 > 172.20.201.198.2050: [udp sum ok] udp 19
14:10:19.205504 0:50:56:40:0:6d 0:3:47:8c:89:c2 ip 89: IP (tos 0xc0, ttl 253,
id 42602, len 75) 172.20.201.198 > 10.10.10.165: icmp 55: 172.20.201.198 udp
port 2051 unreachable for IP (tos 0x0, ttl 126, id 44815, len 47)
10.10.10.165.2022 > 172.20.201.198.2051: [udp sum ok] udp 19
14:10:19.206489 0:50:56:40:0:6d 0:3:47:8c:89:c2 ip 89: IP (tos 0xc0, ttl 253,
id 42603, len 75) 172.20.201.198 > 10.10.10.165: icmp 55: 172.20.201.198 udp
port 2052 unreachable for IP (tos 0x0, ttl 126, id 44816, len 47)
10.10.10.165.2022 > 172.20.201.198.2052: [udp sum ok] udp 19
.....
14:10:59.817571 0:3:47:8c:89:c2 0:50:56:40:0:6d ip 63: IP (tos 0x0, ttl 128,
id 50028, len 49) 10.10.10.165.2084 > 172.20.201.198.161: [udp sum ok]
[len19<asnlen68]
14:10:59.817612 0:3:47:8c:89:c2 0:50:56:40:0:6d ip 63: IP (tos 0x0, ttl 128,
id 50029, len 49) 10.10.10.165.2084 > 172.20.201.198.162: [udp sum ok]
[len19<asnlen68]
.....
14:11:35.813389 0:3:47:8c:89:c2 0:50:56:40:0:6d ip 63: IP (tos 0x0, ttl 128,
id 62178, len 49) 10.10.10.165.2084 > 172.20.201.198.161: [udp sum ok]
[len19<asnlen68]

```

Here is my speculation about network topology:

```

              (A)                                (B)
Attacker (10.10.10.165) |G (10.10.10.1)-| Target Host (172.20.201.198)
                        IDS

```

A: Ethernet address [0:3:47:8c:89:c2] - Intel Corp.
B: Ethernet address [0:50:56:40:0:6d] - VmWare Inc.

I have used the following website: <http://standards.ieee.org/regauth/oui/index.shtml> to find out network card manufacturer for these hosts.

NOTE:

I have used 'A' for Attacker host and 'B' for target host for our discussion.


```
[**] [1:1443:2] TFTP GET passwd [**]  
[Classification: Successful Administrator Privilege Gain] [Priority: 1]  
11/18/03-14:11:35.813389 10.10.10.165:2084 -> 172.20.201.198:161 UDP TTL:128  
TOS:0x0 ID:62178 IpLen:20 DgmLen:49  
Len: 21
```

```

alert udp any any -> any 69 (msg:"TFTP GET passwd"; content: "|0001|"; offset:0;
depth:2; content:"passwd"; offset:2; nocase; classtype:successful-admin; sid:1443;
rev:2;)
alert udp $EXTERNAL_NET any -> $HOME_NET 161 (msg:"SNMP request udp";
reference:cve,CAN-2002-0012; reference:cve,CAN-2002-0013; sid:1417; rev:2;
classtype:attempted-recon;)
alert udp $EXTERNAL_NET any -> $HOME_NET 162 (msg:"SNMP trap udp";
reference:cve,CAN-2002-0012; reference:cve,CAN-2002-0013; sid:1419; rev:2;
classtype:attempted-recon;)

```

Here is a some example of traffic that has not generated false alerts.

[illegible]


```

0x0040: 00 EC 00 1B 8A FE 55 44 50 20 53 63 61 6E 20 62  ....UDP Scan b
0x0050: 79 20 49 53 53 20 28 31 29                      y ISS (1)

```

```

=====

```

So now in this above example attacker A is scanning for UDP port(s), if the port is closed on the target machine then it will generate an ICMP destination Unreachable (port unreachable) message. This is an example of passively fingerprinting target system.

3) PROBABILITY THE SOURCE ADDRESS WAS SPOOFED

UDP is known to be very easy to spoof because of connection-less protocol. Sender will not receive any acknowledgement from receiver so there is a possibility for IP address spoofing. In this case the attacker is in reconnaissance stage and trying to determine which port is listening on remote system. So attacker has been waiting for a response such as, port unreachable icmp message for passively figuring out the remote host's Operating System.

The windump or tcpdump also run to find out if IP address is really spoofed.
C:\>windump -r c:\2003.12.15.8 -e -n -v host 10.10.10.165 >> c:\spoof-test.txt

-r = file to read
-e = print Ethernet header
-n = not convert address to name

```

14:10:14.171828 0:3:47:8c:89:c2 0:50:56:40:0:6d 0800 61: IP (tos 0x0, ttl
128, id 44697, len 47) 10.10.10.165.2022 > 172.20.201.198.236:  udp 19
14:10:14.171898 0:3:47:8c:89:c2 0:50:56:40:0:6d 0800 61: IP (tos 0x0, ttl
128, id 44698, len 47) 10.10.10.165.2022 > 172.20.201.198.237:  udp 19
14:10:14.171942 0:3:47:8c:89:c2 0:50:56:40:0:6d 0800 61: IP (tos 0x0, ttl
128, id 44699, len 47) 10.10.10.165.2022 > 172.20.201.198.238:  udp 19
14:10:14.171982 0:3:47:8c:89:c2 0:50:56:40:0:6d 0800 61: IP (tos 0x0, ttl
128, id 44700, len 47) 10.10.10.165.2022 > 172.20.201.198.239:  udp 19

```

Looks to be same Ethernet address is associated with same IP address.

I have run 'pof' passive fingerprinting tool to find out operating systems involved. I was able to see attacker may be using a Windows machine. The program pof does not find much information about target host.

Pof -s c:\2003.12.15.8 -o c:\gcia.txt

```

Wed Jan 07 16:55:41 2004> 10.10.10.165:2034 - Windows 2000 SP2+, XP SP1
.....

```

4) DESCRIPTION OF ATTACK

UDP is a connectionless protocol defined in RFC 792. Since this protocol is connectionless it is hard to make secure. Many firewalls do not block UDP traffic to perimeter host. There is CERT advisory issue CA-1996-001 for UDP port denial-of-service Attack. Scanner may create a "UDP Packet storm" which cause host to perform slow. The attacker might be looking for any Trojan/backdoor which listen on High UDP ports. For Example Deep Throat (2140), Back orifice 2000(54321) or Hack'a'Tack (31789). Now Most network administrator block ICMP to their network so UDP Scanning is getting more popular for fingerprinting. There are tools available today such as NMAP when running -sU option or Hping. But in this case the attacker has used a Commercial product "ISS Internet Scanner".

5) ATTACK MECHANISM

I have used windump and ethereal v. 0.9.15 to investigate in detail.

C:\>windump -r c:\2003.12.15.8 -v host 10.10.10.165 and host 172.20.201.198

```
14:10:14.171828 IP (tos 0x0, ttl 128, id 44697, len 47) 10.10.10.165.2022 >
172.20.201.198.236: [udp sum ok] udp 19
14:10:14.171898 IP (tos 0x0, ttl 128, id 44698, len 47) 10.10.10.165.2022 >
172.20.201.198.237: [udp sum ok] udp 19
14:10:14.171942 IP (tos 0x0, ttl 128, id 44699, len 47) 10.10.10.165.2022 >
172.20.201.198.238: [udp sum ok] udp 19
....
14:10:14.196294 IP (tos 0xc0, ttl 253, id 42550, len 75) 172.20.201.198 >
10.10.10.165: icmp 55: 172.20.201.198 udp port 236 unreachable
14:10:14.198843 IP (tos 0xc0, ttl 253, id 42551, len 75) 172.20.201.198 >
10.10.10.165: icmp 55: 172.20.201.198 udp port 237 unreachable
14:10:14.200707 IP (tos 0xc0, ttl 253, id 42552, len 75) 172.20.201.198 >
10.10.10.165: icmp 55: 172.20.201.198 udp port 238 unreachable
....
14:10:19.173617 IP (tos 0x0, ttl 128, id 44814, len 47) 10.10.10.165.2022 >
172.20.201.198.2050: [udp sum ok] udp 19 14:10:19.173680 IP (tos 0x0, ttl
128, id 44815, len 47) 10.10.10.165.2022 > 172.20.201.198.2051: [udp sum ok]
udp 19
14:10:19.173738 IP (tos 0x0, ttl 128, id 44816, len 47) 10.10.10.165.2022 >
172.20.201.198.2052: [udp sum ok] udp 19
....
14:10:19.201202 IP (tos 0xc0, ttl 253, id 42601, len 75) 172.20.201.198 >
10.10.10.165: icmp 55: 172.20.201.198 udp port 2050 unreachable
14:10:19.205504 IP (tos 0xc0, ttl 253, id 42602, len 75) 172.20.201.198 >
10.10.10.165: icmp 55: 172.20.201.198 udp port 2051 unreachable
....
14:10:24.175391 IP (tos 0x0, ttl 128, id 44939, len 47) 10.10.10.165.2022 >
172.20.201.198.2055: [udp sum ok] udp 19
14:10:24.175453 IP (tos 0x0, ttl 128, id 44940, len 47) 10.10.10.165.2022 >
172.20.201.198.2056: [udp sum ok] udp 19
14:10:24.175496 IP (tos 0x0, ttl 128, id 44941, len 47) 10.10.10.165.2022 >
172.20.201.198.2057: [udp sum ok] udp 19
...
```

```

14:10:24.199865 IP (tos 0xc0, ttl 253, id 42681, len 75) 172.20.201.198 >
10.10.10.165: icmp 55: 172.20.201.198 udp port 2055 unreachable
14:10:24.200023 IP (tos 0xc0, ttl 253, id 42682, len 75) 172.20.201.198 >
10.10.10.165: icmp 55: 172.20.201.198 udp port 2056 unreachable
14:10:24.203752 IP (tos 0xc0, ttl 253, id 42683, len 75) 172.20.201.198 >
10.10.10.165: icmp 55: 172.20.201.198 udp port 2057 unreachable
.....

```

You see in the above example the attacker is scanning for UDP ports. In above scan attempt you can see ttl values and source ports are not changing but IP ID and destination ports are changing means attacker is initiating new connections. You also see icmp unreachable message for some ports which are closed. One more thing I noticed is the timestamp. Scanner is trying to scan three times every 5 minutes because of its connectionless nature of UDP. Scanner may assume packets may not reach the target so it will try a couple more times to make sure that port is really closed.

```

14:10:14.171828 IP (tos 0x0, ttl 128, id 44697, len 47) 10.10.10.165.2022 >
172.20.201.198.236: [udp sum ok] udp 19
      4500 002f ae99 0000 8011 019b 0a0a 0aa5
      ----- IP Header
      ac14 c9c6 07e6 00ec 001b 8afe 5544 5020
      -----| UDP Header-----|-----data
      5363 616e 2062 7920 4953 5320 2831 29
      -----data -----
14:10:14.171898 IP (tos 0x0, ttl 128, id 44698, len 47) 10.10.10.165.2022 >
172.20.201.198.237: [udp sum ok] udp 19
      4500 002f ae9a 0000 8011 019a 0a0a 0aa5
      ac14 c9c6 07e6 00ed 001b 8afc 5544 5020
      5363 616e 2062 7920 4953 5320 2832 29
14:10:14.171942 IP (tos 0x0, ttl 128, id 44699, len 47) 10.10.10.165.2022 >
172.20.201.198.238: [udp sum ok] udp 19
      4500 002f ae9b 0000 8011 0199 0a0a 0aa5
      ac14 c9c6 07e6 00ee 001b 8afa 5544 5020
      5363 616e 2062 7920 4953 5320 2833 29 .....n
14:10:14.196294 IP (tos 0xc0, ttl 253, id 42550, len 75) 172.20.201.198 >
10.10.10.165: icmp 55: 172.20.201.198 udp port 236 unreachable
      45c0 004b a636 0000 fd01 8c31 ac14 c9c6
      -----IP Header-----
      0a0a 0aa5 0303 87b3 0000 0000 4500 002f
      -----|---icmp -----
      ae99 0000 7e11 039b 0a0a 0aa5 ac14 c9c6
      07e6 00ec 001b 8afe 5544 5020 5363 616e
      2062 7920 4953 5320 2831 29
.....n

```

You can see in above packets all length fields looks to be right. Only in the data field did ISS craft custom data. From all the above details I can conclude the attacker is in the first stage of reconnaissance. The attacker is trying to determine open UDP ports or looking for any backdoor application which would allow remote access to target.

6) CORRELATIONS

This type of UDP scan is not new at this time. NMAP also has an option to use UDP scan for fingerprinting. Since ISS is crafting some data which makes them different than other scanners. Even now ISS scanner has an option for smart UDP scan which reduce false positive scan. I could not find any snort signature for this kind of UDP scan.

Find knowledgebase article at ISS website <http://xforce.iss.net/xforce/xfdb/3105> about UDP Scan.

Here some more articles about UDP Denial-of service attack.

<http://www.securityfocus.com/bid/4111>

<http://xforce.iss.net/xforce/xfdb/8572>

<http://www.cert.org/advisories/CA-1996-01.html>

7) EVIDENCE OF ACTIVE TARGETING

```
14:10:14.171828 IP (tos 0x0, ttl 128, id 44697, len 47) 10.10.10.165.2022 >
172.20.201.198.236: [udp sum ok] udp 19
14:10:14.171898 IP (tos 0x0, ttl 128, id 44698, len 47) 10.10.10.165.2022 >
172.20.201.198.237: [udp sum ok] udp 19 .....
14:11:41.018283 IP (tos 0x0, ttl 128, id 63474, len 49) 10.10.10.165.2074 >
172.20.201.1.183: [udp sum ok] udp 21
14:11:41.018345 IP (tos 0x0, ttl 128, id 63475, len 49) 10.10.10.165.2074 >
172.20.201.1.184: [udp sum ok] udp 21 ....
14:11:42.167979 IP (tos 0x0, ttl 128, id 63828, len 48) 10.10.10.165.2073 >
172.20.201.135.89: [udp sum ok] udp 20
14:11:42.168022 IP (tos 0x0, ttl 128, id 63829, len 48) 10.10.10.165.2073 >
172.20.201.135.90: [udp sum ok] udp 20 .....
```

As we can see above, the attacker has launched a scan on many hosts on the network. The attacker was performing a comprehensive scan on all hosts to find out live hosts that could be a target.

8) SEVERITY

To calculate severity of these alerts I have followed the guidelines which available at http://www.giac.org/GCIA_assignment.php.

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

Criticality: measure of how critical the target system is.

In the log I see ip port unreachable message for almost all port that scanner was scanning for this host so that means appears to be all scanned ports are closed.

Criticality=0

Lethality: how severe the damage to targeted system would be if attack occurred.

Looks to be attacker is doing reconnaissance at this point. It is safe for me to assume at this point 2, but in reality this type of scan is common so we can give 1.

System Countermeasures: measure of the strength of defensive mechanisms in place on host itself.

Again appears to be all scanned port closed so I can assume 5.

Network Counter Measures: measure of defensive mechanisms in place on network such as firewall.

We can see ports were closed but attacker is getting ip unreachable messages. Appears to be router do not have any access-list to protect against this kind of scan. I can assume to be 4.

Severity= (0+2)-(5+4) = -7

These ratings do not require any further more investigation at this stage.

9) DEFENSIVE RECOMMENDATION

Block unnecessary UDP ports for all incoming UDP traffic at firewall. Setup the router to not send IP unreachable message to sender. For Cisco you can do by "no ip unreachable" command. Block incoming and outgoing icmp message. Honeypots are also a good defensive for UDP port scanner. Also do not forget host hardening is very important, always keep updating the operating system for newer vulnerability and disable all unnecessary services.

10) Multiple choice question

1) If you send large packets (more than MTU), packet will get fragmented at router and destination host will re-assemble this packets. What happened if one packet will miss from same fragmentation train?

- A) Destination host send message for resend packet that lost
- B) No big attempt to reassemble and the whole IP packet will be dropped.
- C) Generate IP unreachable message

Answer: B

UDP does not do something as TCP because of its connection-less nature. It will do not make any attempt to reassemble this and just drops whole IP packet.

Assignment: 3: Analyze This

Executive Summary:

Over the Five-day period, there were about 40,446 alert, 7,911,236 scanning probes and 4,959 out-of-spec alerts reported. The following graphs show better trends by hours for alerts and scanning probes. To keep a manageable size of the alerts, I have removed all port scan entries from the alert log files. Technically those entries must also be in port scan logs so I do not want to analyze it twice.

Alerts by Hours of day:

Hour of day	Events	Events bar
midnight - 1:00 AM	1,225	
1:00 AM - 2:00 AM	920	
2:00 AM - 3:00 AM	571	
3:00 AM - 4:00 AM	1,361	
4:00 AM - 5:00 AM	204	
5:00 AM - 6:00 AM	332	
6:00 AM - 7:00 AM	3,112	
7:00 AM - 8:00 AM	1,917	
8:00 AM - 9:00 AM	2,690	
9:00 AM - 10:00 AM	1,629	
10:00 AM - 11:00 AM	1,978	
11:00 AM - noon	1,048	
noon - 1:00 PM	740	
1:00 PM - 2:00 PM	1,553	
2:00 PM - 3:00 PM	687	
3:00 PM - 4:00 PM	709	
4:00 PM - 5:00 PM	1,477	
5:00 PM - 6:00 PM	3,217	
6:00 PM - 7:00 PM	1,117	
7:00 PM - 8:00 PM	672	
8:00 PM - 9:00 PM	2,867	
9:00 PM - 10:00 PM	1,206	
10:00 PM - 11:00 PM	3,904	
11:00 PM - midnight	5,310	
Average	1,685	
Total	40,446	

Scans by Hours of day:

Hour of day	Hits	Hits bar
midnight - 1:00 AM	327,542	
1:00 AM - 2:00 AM	362,812	
2:00 AM - 3:00 AM	304,110	
3:00 AM - 4:00 AM	356,047	
4:00 AM - 5:00 AM	301,970	
5:00 AM - 6:00 AM	318,521	
6:00 AM - 7:00 AM	523,095	
7:00 AM - 8:00 AM	476,599	
8:00 AM - 9:00 AM	495,791	
9:00 AM - 10:00 AM	439,086	
10:00 AM - 11:00 AM	355,447	
11:00 AM - noon	404,818	
noon - 1:00 PM	337,909	
1:00 PM - 2:00 PM	296,892	
2:00 PM - 3:00 PM	297,514	
3:00 PM - 4:00 PM	293,169	
4:00 PM - 5:00 PM	241,600	
5:00 PM - 6:00 PM	212,980	
6:00 PM - 7:00 PM	253,417	
7:00 PM - 8:00 PM	251,590	
8:00 PM - 9:00 PM	265,693	
9:00 PM - 10:00 PM	236,028	
10:00 PM - 11:00 PM	290,170	
11:00 PM - midnight	268,436	
Average	329,634	
Total	7,911,236	

The University was using mostly customized snort rules for detections but still there are some configuration issues resulting some false alerts. Another major issue is the tremendous amount of peer-to-peer (P2P) file sharing traffic on the network. The file sharing on a P2P network with external hosts exposes the university networks to viruses and Trojans.

Aside from some false alarms and less important alerts, some internal hosts need further investigation. These hosts should be taken offline and investigated by technical staff.

MY.NET.97.22
MY.NET.97.66
MY.NET.97.193
MY.NET.97.179
MY.NET.97.149
MY.NET.97.81
MY.NET.97.178

Logs Analyzed:

The Logs files used for this analysis were from the time period Feb. 4th, 2004 through Feb. 8th, 2004. These files were downloaded from <http://www.incidents.org/logs/>.

The specific details are as follows:

Alerts	Scan	OOS (Out of Spec)
Alert.040204.gz	Scans.040204.gz	Oos_report_040204
Alert.040205.gz	Scans.040205.gz	Oos_report_040205
Alert.040206.gz	Scans.040206.gz	Oos_report_040206
Alert.040207.gz	Scans.040207.gz	Oos_report_040207
Alert.040208.gz	Scans.040208.gz	Oos_report_040208

Without knowing information about the monitored network, some assumptions have been made in order for further analysis. The logs were generated with Snort IDS systems. Since Snort version is not specified, I will assume the current version was used with slightly modified rule sets.

Summary of Alerts:

Below is a list of all alerts generated in a five-day period. They are ordered according to the number of alerts generated per signature. Using snortsnarf-analyzed data generates this summary. It has provided very useful information by providing summarized alerts by signatures, numbers of unique source and destination hosts involved.

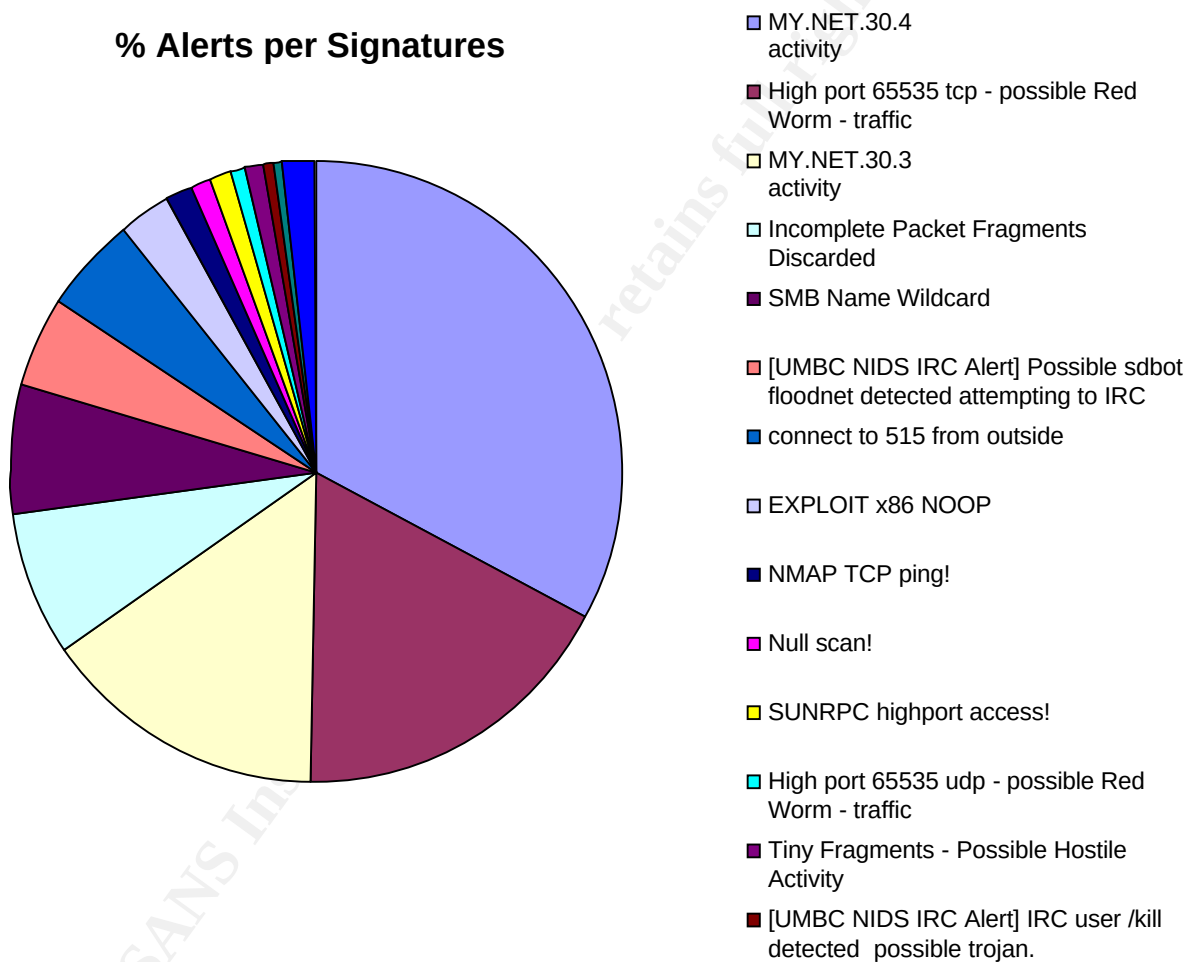
Summary of all Snort alerts between Feb. 4th, 2004 and Feb. 8th, 2004 by no. alerts per signature:

Signature	Alerts	Sources	Dest.
MY.NET.30.4 activity	13240	282	1
High port 65535 tcp - possible Red Worm - traffic	7140	101	135
MY.NET.30.3 activity	5971	121	1
Incomplete Packet Fragments Discarded	3137	59	39
SMB Name Wildcard	2653	53	373
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	1949	8	3
connect to 515 from outside	1946	1	1
EXPLOIT x86 NOOP	1150	239	90

NMAP TCP ping!	579	140	48
Null scan!	489	93	52
SUNRPC highport access!	415	25	28
High port 65535 udp - possible Red Worm - traffic	334	52	49
Tiny Fragments - Possible Hostile Activity	316	6	7
[UMBC NIDS IRC Alert] IRC user /kill detected possible trojan.	256	47	50
Possible trojan server activity	193	38	37
TCP SRC and DST outside network	128	34	50
IRC evil - running XDCC	127	9	7
RFB - Possible WinVNC - 010708-1	53	12	13
FTP passwd attempt	51	45	1
[UMBC NIDS] External MiMail alert	46	17	1
SMB C access	46	24	3
TCP SMTP Source Port traffic	38	3	6
EXPLOIT x86 setuid 0	32	26	23
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	30	4	6
EXPLOIT x86 setgid 0	29	24	23
ICMP SRC and DST outside network	26	18	23
EXPLOIT NTPDX buffer overflow	8	6	6
TFTP - Internal UDP connection to external tftp server	8	3	3
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	8	4	1
DDOS shaft client to handler	6	2	1
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	6	2	2
SYN-FIN scan!	5	5	4
EXPLOIT x86 stealth noop	4	3	3
[UMBC NIDS IRC Alert] K\line'd user detected possible trojan.	3	1	3
External FTP to HelpDesk MY.NET.70.49	2	2	1
FTP DoS ftpd globbing	2	2	2
Attempted Sun RPC high port access	2	1	1
External FTP to HelpDesk MY.NET.70.50	2	2	1
NIMDA - Attempt to execute cmd from campus host	2	2	2
TFTP - External UDP connection to internal tftp server	1	1	1
Probable NMAP fingerprint attempt	1	1	1
MY.NET.30.4 activity [**] 68.33.138.19302/06-17:30:52.166475 [**] MY.NET.30.4 activity	1	1	1
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC [**] MY.NET.97.4002/08-07:41:47.803783 [**] [UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	1	1	1
Back Orifice	1	1	1
High port 65535 tcp - possible Red Worm - traffic [**] 220.12.4.21702/07-23:16:04.676946 [**] High port 65535 tcp - possible Red Worm - traffic	1	1	1
MY.NET.30.4 activity [**] 129.41.68.202/05-14:17:51.927120 [**] Null scan!	1	1	1
High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.42.102/07-23:09:12.210720 [**] High port 65535 tcp - possible Red Worm - traffic	1	1	1
High port 65535 tcp - possible Red Worm - traffic [**] 220.12.4.21702/07-	1	1	1

23:45:01.992508 [**] High port 65535 tcp - possible Red Worm - traffic			
[UMBC NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot	1	1	1
EXPLOIT x86 NOPS	1	1	1
NETBIOS NT NULL session	1	1	1
DDOS mstream client to handler	1	1	1
External FTP to HelpDesk MY.NET.53.29	1	1	1

% Alerts per Signatures



Over 40446 events were recorded in five days of alerts data. Some Snort rules were modified from the defaults for their needs. There is some evidence of some configuration errors, which lead to false positive Snort alerts. I will be discussing those in later sections. My goal in this analysis is to find out noisy rules, which may hide real intrusive alerts.

Top 20 most active sources IP involved:

Total # Alerts	Source IP	# Signatures triggered	Destinations involved
3097 alerts	220.12.4.217	5 signatures	MY.NET.42.1
3015 alerts	MY.NET.42.1	4 signatures	(5 destination IPs)
1972 alerts	68.33.138.193	2 signatures	MY.NET.30.4
1946 alerts	68.32.127.158	1 signatures	MY.NET.24.15
1751 alerts	68.55.241.46	1 signatures	MY.NET.30.4
1659 alerts	MY.NET.75.13	1 signatures	(155 destination IPs)
1642 alerts	MY.NET.97.40	2 signatures	194.68.45.50
1434 alerts	68.86.184.21	1 signatures	MY.NET.30.4
1300 alerts	131.92.177.18	1 signatures	MY.NET.30.3
1138 alerts	MY.NET.21.68	1 signatures	(6 destination IPs)
1123 alerts	129.41.68.2	2 signatures	MY.NET.30.3, MY.NET.30.4
1032 alerts	68.50.102.64	1 signatures	MY.NET.30.4
952 alerts	68.55.27.157	2 signatures	MY.NET.30.3, MY.NET.30.4
890 alerts	MY.NET.21.67	1 signatures	(6 destination IPs)
862 alerts	MY.NET.21.69	1 signatures	(4 destination IPs)
779 alerts	24.35.70.49	1 signatures	MY.NET.30.4
670 alerts	68.54.168.204	1 signatures	MY.NET.30.4
653 alerts	68.55.241.230	1 signatures	MY.NET.30.4
648 alerts	68.55.178.168	2 signatures	MY.NET.30.3, MY.NET.30.4
501 alerts	68.55.62.79	2 signatures	MY.NET.30.3, MY.NET.30.4

Top 20 active destinations IP involved:

Total # Alerts	Destination IP	# Signatures triggered	Originating sources
13242 alerts	MY.NET.30.4	3 signatures	(283 source IPs)
5971 alerts	MY.NET.30.3	1 signatures	(121 source IPs)
3138 alerts	MY.NET.42.1	10 signatures	(15 source IPs)
2956 alerts	220.12.4.217	2 signatures	MY.NET.42.1
1946 alerts	MY.NET.24.15	1 signatures	68.32.127.158
1942 alerts	194.68.45.50	2 signatures	(3 source IPs)
1058 alerts	65.168.38.242	1 signatures	MY.NET.75.13
979 alerts	65.141.62.206	1 signatures	(3 source IPs)
870 alerts	68.10.226.197	1 signatures	(3 source IPs)
482 alerts	208.155.109.200	1 signatures	MY.NET.21.68, MY.NET.21.67
327 alerts	MY.NET.1.3	2 signatures	(50 source IPs)
315 alerts	202.91.34.9	1 signatures	(3 source IPs)
250 alerts	MY.NET.98.19	1 signatures	221.189.158.241
222 alerts	MY.NET.98.62	1 signatures	66.187.232.35
220 alerts	221.189.158.241	1 signatures	MY.NET.98.19
219 alerts	MY.NET.12.6	9 signatures	(74 source IPs)
218 alerts	MY.NET.82.87	3 signatures	(3 source IPs)
204 alerts	204.152.186.58	1 signatures	(3 source IPs)
164 alerts	MY.NET.5.20	2 signatures	(4 source IPs)
133 alerts	MY.NET.153.37	7 signatures	(13 source IPs)

Frequent Alert Details:

MY.NET.30.4 Activity

Severity: Noise

Reported: 13,240 times

Background:

This is a custom alert referring to MY.NET.30.4 (MY.NET.30.4). Most traffic is destined for ports 80, 51443, 524. Some Novell Network Implementations may generate this traffic. The Novell apache service iFolder use port 51442 by default. May be it is used for sharing files over the Internet.

Analysis:

02/06-17:18:39.18 [**] MY.NET.30.4 activity [**] 68.33.138.193:15197 -> MY.NET.30.4:80
02/06-17:18:39.20 [**] MY.NET.30.4 activity [**] 68.33.138.193:15196 -> MY.NET.30.4:80
02/06-17:18:49.08 [**] MY.NET.30.4 activity [**] 68.33.138.193:15207 ->
MY.NET.30.4:51443
02/06-17:18:49.46 [**] MY.NET.30.4 activity [**] 68.33.138.193:15207 ->
MY.NET.30.4:51443
...
02/06-06:10:25.22 [**] MY.NET.30.4 activity [**] 68.55.241.46:1224 -> MY.NET.30.4:51443
02/06-06:10:25.23 [**] MY.NET.30.4 activity [**] 68.55.241.46:1224 -> MY.NET.30.4:51443
02/06-06:10:25.23 [**] MY.NET.30.4 activity [**] 68.55.241.46:1222 -> MY.NET.30.4:51443
...
02/04-08:24:40.16 [**] MY.NET.30.4 activity [**] 129.41.68.2:57001 -> MY.NET.30.4:524
02/04-08:24:40.18 [**] MY.NET.30.4 activity [**] 129.41.68.2:57001 -> MY.NET.30.4:524
02/04-08:25:40.30 [**] MY.NET.30.4 activity [**] 129.41.68.2:57001 -> MY.NET.30.4:524
...
02/08-13:48:28.58 [**] MY.NET.30.4 activity [**] 66.44.63.88:1312 -> MY.NET.30.4:80
02/08-13:48:41.03 [**] MY.NET.30.4 activity [**] 66.44.63.88:1312 -> MY.NET.30.4:80

Correlations:

<http://support.novell.com/cgi-bin/search/searchtid.cgi?/10078035.htm>

<http://support.novell.com/cgi-bin/search/searchtid.cgi?/10067181.htm>

http://www.giac.org/practical/GCIA/Pete_Storm_GCIA.pdf

High port 65535 tcp - possible Red Worm – traffic

Severity: Noise

Reported: 7140 times

Background:

Red Worm (Adore worm) is similar to the lion and ramen worms. It is a Linux worm, which uses vulnerability in older versions of wu-ftpd, LPrng, rpc-statd and BIND. A nice article on this worm can be found at http://www.dials.ru/english/inf/linux_adore.htm. The end result of this worm is the opening of port 65535 which waits for connections and initiates a root shell session with remote shell.

Analysis:

There were 7140 alerts for port 65535. The table below is a list of top source and destination, with a number of destinations and sources involved. Host 220.12.4.217 and MY.NET.42.1 were generating the most percentage of alerts.

Source	# Alerts (total)	# Dsts (total)
220.12.4.217	3097	1
MY.NET.42.1	3015	5
221.189.158.241	250	1
MY.NET.98.19	220	1
MY.NET.34.11	55	6
MY.NET.34.14	36	12
MY.NET.53.60	27	1
24.215.192.158	27	1

Destinations	# Alerts (total)	# Srcs (total)
MY.NET.42.1	3138	15
220.12.4.217	2956	1
MY.NET.98.19	250	1
221.189.158.241	220	1
24.215.192.158	41	1
MY.NET.34.11	70	27
MY.NET.12.6	219	74
203.165.202.88	27	1

Count	Src ports -> Dst Posts
3092	65535 -> 4662
2955	80 -> 65535
250	4662 -> 65535
220	65535 -> 80
199	2025 -> 65535
111	65535 -> 3472
83	113 -> 65535
66	110 -> 65535
27	65535 -> 110

Another rule appears to be triggering for Source or destination TCP port 65535. This may lead to many false positive alerts. There is definitely a correlation in the above table's data. Port 4662 is involved in some peer file sharing. I believe this is a false positive alert for some peer file sharing traffic. Port 65535 source or destination ports have triggered these alerts.

MY.NET.30.3 activity

Severity: Noise

Reported: 5971 times

Background:

This is also another custom alert. Most traffic was destined for MY.NET.30.3 host for TCP Port 524. The traffic may be generated by Novell Netware systems. In pure IP Mode and not using SLP for locating servers, all communications happen on this port.

Analysis:

```
02/04-08:24:40.164151  [**] MY.NET.30.4 activity [**] 129.41.68.2:57001 ->
MY.NET.30.4:524
02/04-08:24:40.189013  [**] MY.NET.30.4 activity [**] 129.41.68.2:57001 ->
MY.NET.30.4:524
02/04-08:25:40.305980  [**] MY.NET.30.4 activity [**] 129.41.68.2:57001 ->
MY.NET.30.4:524
02/04-08:25:40.558755  [**] MY.NET.30.4 activity [**] 129.41.68.2:57001 ->
MY.NET.30.4:524
```

Correlations:

<http://support.novell.com/cgi-bin/search/searchtid.cgi?/10014320.htm>
<http://support.novell.com/cgi-bin/search/searchtid.cgi?/10013531.htm>

Incomplete Packet Fragments Discarded

Severity: Noise

Reported: 3137 times

Snort Signature ID: None

These alerts were more of a snort configuration issue. They come from an old defragmentation preprocessor. The newer frag2 preprocessor should be used instead. According to Dragos, Raju, this message is given by the defragmentation preprocessor when packets bigger than 8k that are more than half empty when the last fragment is received are discarded. This may also be caused by transmission errors.

Analysis:

```
02/04-20:53:02.656025  [**] Incomplete Packet Fragments Discarded [**]
MY.NET.21.68 -> 68.10.226.197
02/04-20:53:02.911583  [**] Incomplete Packet Fragments Discarded [**]
MY.NET.21.68 -> 68.10.226.197
02/04-20:53:05.706659  [**] Incomplete Packet Fragments Discarded [**]
MY.NET.21.68 -> 68.10.226.197
....
2/04-16:35:07.695584  [**] Incomplete Packet Fragments Discarded [**]
209.208.199.46:0 -> MY.NET.152.12:0
02/04-16:35:08.793386  [**] Incomplete Packet Fragments Discarded [**]
209.208.199.46:0 -> MY.NET.152.12:0
```

Last example shows traffics to source and destination port 0, which is a reserved port. We should not see any normal traffics using same port in any directions.

Correlations:

<http://www.geocrawler.com/archives/3/4890/2001/2/350/5151528>

Cut, Sed, Awk, Sort

SMB Name Wildcard

Severity: Noise

Reported: 2653 times

Snort Signature ID: ?

alert UDP \$EXTERNAL any -> \$INTERNAL 137 (msg: "IDS177/netbios-name-query";
content: "CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA|00 00|";)

Background:

Server Message Block (SMB) protocol is used when File and print sharing is turned on windows systems. The windows systems used these types of queries in normal operations to locate the NetBIOS name when the IP address is known. This types of traffic should not be allowed from outside. There is also worm activity (network.vbs) involved with this port.

Analysis:

The following tables show all source hosts involved in this traffic. I have only found internal hosts initiating all traffic. This is definitely noise for IDS Systems. These kinds of alerts may hide some real attacks from an intrusion analyst. I do not see any SMB alerts from outside so we can safely ignore these alerts.

Source	# Alerts (total)
MY.NET.75.13	1659
MY.NET.150.198	231
MY.NET.150.44	178
MY.NET.190.102	86
MY.NET.53.110	74
MY.NET.150.11	65
MY.NET.11.4	61
MY.NET.112.161	54
MY.NET.42.3	59
MY.NET.42.2	26
MY.NET.53.223	15
MY.NET.190.95	15
MY.NET.190.92	14
MY.NET.42.1	3015
MY.NET.190.93	13
MY.NET.29.30	15
MY.NET.42.6	10
MY.NET.153.196	9
MY.NET.53.50	8
MY.NET.153.21	7
MY.NET.153.93	5
MY.NET.153.77	5
MY.NET.62.2	5

Source	# Alerts (total)
MY.NET.42.4	9
MY.NET.152.168	2
MY.NET.84.216	2
MY.NET.53.31	2
MY.NET.153.153	5
MY.NET.153.145	2
MY.NET.75.20	2
MY.NET.66.50	2
MY.NET.53.61	2
MY.NET.75.155	2
MY.NET.153.90	1
MY.NET.81.81	1
MY.NET.153.157	1
MY.NET.153.182	1
MY.NET.42.7	85
MY.NET.153.94	1
MY.NET.84.131	1
MY.NET.151.72	1
MY.NET.53.93	1
MY.NET.69.253	1
MY.NET.153.159	4
MY.NET.189.41	1
MY.NET.11.6	1

MY.NET.82.27	4	MY.NET.153.167	1
MY.NET.53.185	4		
MY.NET.84.155	4		
MY.NET.153.151	3		
MY.NET.29.3	3		
MY.NET.109.86	3		

Correlations:

http://whitehats.com/cgi/arachNIDS/Show?_id=ids177&view=research
http://www.sans.org/resources/idaq/port_137.php

[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC

Severity: High

Reported: 1949 times

Snort Signature ID: ?

alert tcp \$HOME_NET any -> \$EXTERNAL_NET 6660:7000 \

(content: "USER "; \

content: " 0 0 "; nocase; \

msg: "Possible sdbot floodnet detected attempting to IRC"; \

classtype:misc-activity;)

Background:

There are 10 different variations of this Trojan. This Trojan connects to IRC channels and accept commands, and is related to Denial of Service attacks and allows remote access to the victim system(s).

Analysis:

The following table shows all internal source hosts involved in this traffic. All these hosts are definitely involved in IRC traffic. Most traffic is resolved to the dal.net IRC Channel. The very few connections to other university may be to share some files through IRC. The Snort Signature only captures logon activity to those ports. It might be possible that these hosts are not compromised but only used for IRC Channels. Still these hosts are a big suspect for SDBOT Trojan and need further investigations.

Source	# Alerts (total)	# Dsts (total)
MY.NET.97.40	1642	1
MY.NET.97.22	298	1
MY.NET.97.66	2	1
MY.NET.97.193	2	1
MY.NET.97.179	2	1
MY.NET.97.149	2	1
MY.NET.97.81	1	1
MY.NET.97.178	1	1

Destinations	# Alerts (total)	# Srcs (total)
194.68.45.50 (irc.dal.net)	1942	3
140.126.17.129 (a17-129.cv.chu.edu.tw)	7	4
169.226.226.247 (H226-247.INDIAN.resnet.albany.edu)	1	1

Correlations:

http://www.giac.org/practical/GCIA/Don_Murdoch_GCIA.pdf
http://www.giac.org/practical/GCIA/Daniel_Clark_GCIA.pdf
<http://www.viruslibrary.com/virusinfo/Backdoor.IRC.SdBot.htm>

Connect to 515 from outside

Severity: Noise

Reported: 1946 times

Snort Signature ID: None

Background:

This port is used for lpd printing service called LPRng. Some vulnerabilities exist for this port for some UNIX versions. See URL <http://www.cert.org/advisories/CA-2000-22.html>.

These alerts were triggered for any traffic destined for port 515.

Analysis:

Connect to 515 from outside [**] 68.32.127.158:51168 -> MY.NET.24.15:515

Source	# Alerts (total)	Destination
68.32.127.158	1946	MY.NET.24.15

We have one source host attacking one destination IP address only. This host needs immediate attention for software version. If this host is running an older version of LPRng then it needs to have a patch applied as soon as possible and block all traffic destined for port 515 at the firewall gateway.

Correlations:

http://www.satx.rr.com/support/security/significant_increase_in_unix.html
<http://xforce.iss.net/xforce/alerts/id/advise80>

IRC evil - running XDCC
Reported: 127 times
Snort Signature ID: None

Background:

This is another custom alert, which triggered alerts for port 6667 or 7000. IRC servers commonly used port 6666, 6667 and 7000. These ports are also used by several Trojans; NetBus, DarkFTP, Moses etc. This Bot provides file-sharing capability to the IRC Interface. You need to connect to the IRC server and associate with a particular channel before you may access resources that XDCC advertises. There is a possibility of a Trojan installed on the Bot, which could look for other host on network to compromise.

The following host MY.NET.42.1, MY.NET.42.3, MY.NET.42.7, MY.NET.42.10, MY.NET.42.8, MY.NET.112.199, MY.NET.42.4, MY.NET.42.11, and MY.NET.82.79 needs immediate attention for further investigations.

TCP SRC and DST outside network

Reported: 124 times
Snort Signature ID: None

Background:

This traffic triggers when IDS sees traffic that is source and destined for an outside network. The internal IP address is neither source nor destination for these packets.

Analysis:

[**] TCP SRC and DST outside network [**] 172.170.100.93:1933 -> 216.155.194.64:5100
[**] TCP SRC and DST outside network [**] 172.170.100.93:5101 -> 67.33.201.6:50935
[**] TCP SRC and DST outside network [**] 172.173.110.24:2312 -> 64.12.39.89:80
[**] TCP SRC and DST outside network [**] 172.173.110.24:2313 -> 64.12.39.67:80

I have looked for this host in scan or OOS logs but it does not seem to exist. This is definitely possible spoofed traffic. We do not have enough information to conclude this for fact. The spoof address is hard to find but not impossible. The TCPDUMP or Ethernet Packet sniffers may help finding these hosts on the local network (depending on network topology).

Recommendations:

Setup ingress and egress access-lists on border routers or firewall to block spoofed traffic.

Scan Analysis:

In this section we will cover the port scan activity that happening on the network.

TOP SOURCE HOST:

Source host	Hits ?	Hits bar
130.85.1.3	2,793,401	
130.85.81.39	1,265,140	
130.85.1.4	460,590	
130.85.153.37	377,484	
130.85.42.6	250,074	
130.85.111.34	245,565	
130.85.163.107	242,314	
130.85.162.92	240,094	
130.85.34.14	215,472	
130.85.153.32	157,926	
2015 other items	1,663,176	
Total	7,911,236	

TOP DESTINATION HOST:

Destination host	Hits ?	Hits bar
69.6.33.10	83,507	
69.6.33.11	75,513	
192.26.92.30	64,808	
203.20.52.5	38,437	
192.5.6.30	38,192	
192.55.83.30	35,335	
192.48.79.30	30,943	
131.118.254.34	30,622	
216.109.116.17	29,134	
131.118.254.33	27,868	
2451249 other items	7,456,877	
Total	7,911,236	

TOP DESTINATION PORTS:

Destination port	Hits ?	Hits bar
53	3,238,147	
135	2,100,946	
6129	379,789	
25	231,217	
80	115,839	
41170	113,323	
20168	73,879	
22321	66,978	
21	40,978	
4899	35,516	
39623 other items	1,514,624	
Total	7,911,236	

For Further Analysis I have broken down Scan Logs by Protocols: TCP and UDP.

TCP Port Scans:

3269397	ALL TCP Scans
3259149	TCP SYN scans
8366	TCP RSV scans
900	TCP INVACK scans
402	TCP NULL scans
260	TCP FIN scans
210	TCP NOACK scans
77	TCP VECNA scans
15	TCP XMAS scans
7	TCP SPAU scans
5	TCP SYNFIN scans
2	TCP NAMPID scans

TCP Port scans with RESERVED BIT:

8366	ALL TCP RSV Scans
7317	TCP RSV SYN scans
313	TCP RSV NOACK scans
301	TCP RSV UNKNOWN scans
235	TCP RSV INVACK scans
63	TCP RSV VECNA scans
54	TCP RSV XMAS scans
24	TCP RSV FIN scans
21	TCP RSV SYNFIN scans
16	TCP RSV NULL scans
14	TCP RSV SPAU scans
8	TCP RSV NMAPID scans

TOP Destination UDP Ports:

Count	Destination ports
3237960	53
113320	41170
66979	22321
13021	123
12479	1214
7091	7674
5220	33492
4992	8402
4867	32912
4593	6112

Scans: Peer-To-Peer Applications

Edonkey Scan:

Top Source	Destinations
130.85.111.34	66.58.18.82
130.85.97.16	218.252.164.20
130.85.81.39	217.172.185.4

Gnutella Scan:

Top Source	Destinations
130.85.98.99	65.33.27.21
130.85.53.225	66.233.191.53
130.85.97.58	24.190.63.202

Kazaa Scan:

Top Source	Destinations
130.85.153.37	209.6.181.179
130.85.111.34	24.44.196.183
130.85.153.85	205.251.242.241
130.85.153.32	81.100.150.135

WinMX Scan:

Top Source
130.85.42.7
130.85.42.1
130.85.42.10

Blubster:

Top Source	Destinations
130.85.72.155	38.118.160.159
130.85.97.37	38.118.160.160
130.85.97.62	65.60.191.2
130.85.97.40	66.31.245.9
130.85.97.224	152.23.186.243

The ports used above are 4665 (Edonkey), 6346 (Gnutella), 6347 (Gnutella), 1214 (Kazaa), 6257 (WinMX), and 41170 (Blubster). The University's network is heavily used by peer-to-peer applications.

PORT 135 ACTIVITY (Worm Propagation):

The vast majority of port 135 activity is certainly indicative of MSBlaster, Welchia or RPC worm propagation attempts.

Count	Source Host
1264580	130.85.81.39
247316	130.85.42.6
242303	130.85.163.107
240089	130.85.162.92
57575	130.85.80.243
21386	130.85.163.234
10518	130.85.150.227

Recommendations:

The port 135 activities are common today on the Internet. Many MSBlaster or Welchia infected hosts still exist on the Internet, which may use up bandwidth on some networks. This needs to be blocked on port 135 at the gateway or some honey pot (Tar pit) which may help for this kind of activity.

Scans for Well-Known Services:

Count	Source Host	Dst. Port
31967	130.85.81.39	21
18452	130.85.111.34	21
19843	209.71.206.162	80
9195	130.115.2.18	80
213574	130.85.34.14	25
32461	130.85.81.39	25

The above host is scanning university networks for ftp, http or SMTP servers. These services more likely to be exposed for inbound traffic.

Port TCP 6129 Probe (Dame Ware Remote Admin):

The External Host is looking for a compromised host, which has Dame Ware remote control tools installed. This port always is in a top ten target ports at Dshield.org.

Count	Top Source
16763	212.7.193.133
14681	211.94.206.191
14107	24.20.215.40
13318	216.191.146.179

Port TCP 20168 Activity (? Windows Update Virus):

Count	Top Source
14989	131.183.184.100
8906	4.19.90.11
8081	166.66.15.149

According to <http://isc.sans.org/diary.html?date=2003-12-11> , this port is attributed to worms which use this port for tftp file transfer of worm code.

OOS Logs:

As a final stage of analysis, we take a closer look at “out-of-spec” packets. The Out of Spec logs are generated when snort encounters a problem with TCP options or flags. This is mainly generated due to some corruption in packets, crafted packets for fingerprinting or a new implementation of ECN (Explicit Congestion Notification) Flag.

TOP 10 OOS SOURCE HOSTS:

Count	Source host
1174	68.54.84.49
521	207.138.63.21
296	207.138.63.20
177	207.138.63.26
156	202.175.245.90
132	207.138.63.25
122	207.138.63.24
92	62.210.155.58
90	67.114.19.186
85	202.138.189.60

The majority traffic from host 68.54.84.49 destined for port 110 and from host 207.138.63.0 networks destined for port 25.

TOP 10 OOS DESTINATIONS:

Count	Destination host
2077	MY.NET.12.6
1320	MY.NET.6.7
325	MY.NET.24.44
145	MY.NET.84.235
103	MY.NET.34.11

96	MY.NET.60.39
70	MY.NET.60.16
70	MY.NET.60.38
66	MY.NET.42.12
55	MY.NET.12.4

It is apparent from the table above that MY.NET seems to be a receiving end of most traffic.

TOP 10 SOURCE PORTS:

Count	Source port
62	3920
31	1088
16	1038
15	80
11	6881
10	2821
9	3656
8	3153
7	1474
7	25

TOP 10 DESTINATION PORTS:

Count	Destination port
2130	25
1229	110
901	80
168	4662
76	113
62	1039
43	6881
42	3300
31	1159
28	2234

The majority of OOS traffic was triggered for two high order ECN flag bits. These flags implement "Active queue management" of data flow to TCP traffic to prevent bottleneck at the router. Many routers have implemented this functionality but many IDS still consider this as a violation of TCP specifications.

```

=====
02/08-00:05:23.401475 216.95.201.20:58437 -> MY.NET.12.6:25
TCP TTL:48 TOS:0x0 ID:51755 IpLen:20 DgmLen:60 DF
12*****S* Seq: 0xD5336D3C Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1380 SackOK TS: 747969960 0 NOP WS: 0
=====

```

Majority of this traffic are false positive. These alerts were triggered because router has set ECN TCP flags.

```

=====
02/08-00:11:08.629907 68.122.128.1:47633 -> MY.NET.12.4:110
TCP TTL:80 TOS:0x0 ID:4660 IpLen:20 DgmLen:40
***** Seq: 0x7D05001 Ack: 0x8D534C97 Win: 0x800 TcpLen: 20
=====

```

=====

[illegible][illegible]

Most of these crafted packets were used in fingerprinting systems. The Out-of-specs logs provide valuable information for crafted packets.

Conclusion and Recommendations:

After analyzing five days worth of all logs files, I believe the university has some snort configuration and rules issues. The IDS rules need to be fine-tuned to further reduce false positive alerts and less important events. Many false alerts always hide real attacks from the intrusion analyst. If the university does not need all preprocessors for snort then they should be turned off in snort configuration.

Security policy is always most important in any network. The university should have established a policy for P2P software use, anti-virus scan etc. The security policy needs to be established for service to be available from outside. There should be an outline established for student access from outside of the network.

Host hardening and patch management are also an important piece in security. Patching all systems on the local network is needed on the university network. Many times the systems administrator will patch important systems but get forget less important systems. (Some printers are also on a UNIX OS with web server for management). The attacker may compromise those systems and then look for other systems to compromise.

I also recommend improving the firewall rules set and access-lists on border gateway routers. The gateway router should be a first line of defense. Setup an access-list to block IP spoofing traffic. Any traffic with a private IP address as a source should not enter from the outside. Establish a content filter to block worm traffic. The honey pot may also help reduce worm and port scan traffic. There should be no traffic allowed on port 135 from the Internet. All port 135 traffic must be blocked by the firewall.

Finally, Security Awareness is also important to teach users about the risk and danger of using P2P applications open window shares, Trojans and legal issues surrounding the sharing of music files, application and other copyright protected information.

Process Used in this analysis:

First I have condensed all five days of logs files into one by using cat and sort commands. I have also found some corruption in the alerts log files. With some manual manipulation and some with scripts, I have cleared up those corruptions. In the alerts and OOS file local network was referenced with MY.NET. to follow the advice from older practical and use some UNIX scripts, sed, awk commands and changed reference MY.NET to some fake address 999.999. I was planning to use ACID for further analysis but it was not worthwhile to write scripts in this short time to put all alerts in a complex database. Then I started looking in other student's assignments as well as their advice. I have tried snortlog but it did not work with these logs. Then I decided to use snortsnarf. I followed the instructions found on their website www.silicondefense.com. The snortsnarf works fine for alert files. The snortsnarf is a memory intensive application. I was using 2Gz processor with 1GB memory but still I was not able to run port scan logs with snortsnarf. I decided to analyze scan logs manually with some UNIX shell scripts using Grep, Awk, Sort, cut and uniq commands.

I have divided the scan logs into TCP and UDP scans for further analysis. I have followed same procedures for OOS logs also. The results were stored in text files and Excel to help organize and sort IP and Port information. I have also tried the sawmill log analyzer to compare my results.

REFERENCES:

www.google.com

<http://www.dshield.org/ipinfo.php>

<http://www.geektools.com/whois.php>

<http://www.silicondefense.com/software/snortsnarf/>

<http://jeremy.chartier.free.fr/snortalog/>

<http://www.snort.org>

<http://www.sawmill.net/>

<http://whitehats.com/>

CERT Advisory CA-2000-22

<http://www.cert.org/advisories/CA-2000-22.html>

http://www.dials.ru/english/inf/linux_adore.htm

<http://www.geocrawler.com/archives/3/4890/2001/2/350/5151528>

<http://ist.uwaterloo.ca/security/vulnerable/20030103.note>

http://www.whitehats.ca/main/members/Herc_Man/Files/Al_Williams_GCIAPractical.pdf

<http://www.ussg.iu.edu/hypermail/linux/kernel/0011.1/1186.html>

http://www.cert.org/current/services_ports.html

<http://cert.uni-stuttgart.de/archive/incidents/2003/06/msg00130.html>

http://www.sans.org/resources/idfaq/port_137.php

http://www.satx.rr.com/support/security/significant_increase_in_unix.html

<http://www.dal.net/admin/vote/index.php3>

<http://www.hping.org/>

<http://support.novell.com/>

<http://www.linklogger.com/TCP6129.htm>

GCIA PRACTICALS:

http://www.giac.org/practical/GCIA/Don_Murdoch_GCIA.pdf

http://www.giac.org/practical/GCIA/Knut_Bjornstad_GCIA.pdf

http://www.giac.org/practical/GCIA/Loic_Juillard_GCIA.pdf

http://www.giac.org/practical/GCIA/Saro_Hayan_GCIA.pdf

http://www.giac.org/practical/GCIA/Marshall_Heilman_GCIA.pdf

http://www.giac.org/practical/GCIA/Pete_Storm_GCIA.pdf

http://www.giac.org/practical/GCIA/Joanne_Schell_GCIA.pdf

http://www.giac.org/practical/GCIA/Johnny_Wong_GCIA.pdf

http://www.giac.org/practical/GCIA/Ian_Martin_GCIA.pdf

http://www.giac.org/practical/GCIA/SaiPrasad_Kesavamatham_GCIA.pdf

http://www.giac.org/practical/GCIA/John_Ruiz_GCIA.pdf

http://www.giac.org/practical/GCIA/Sylvain_Randier_GCIA.pdf

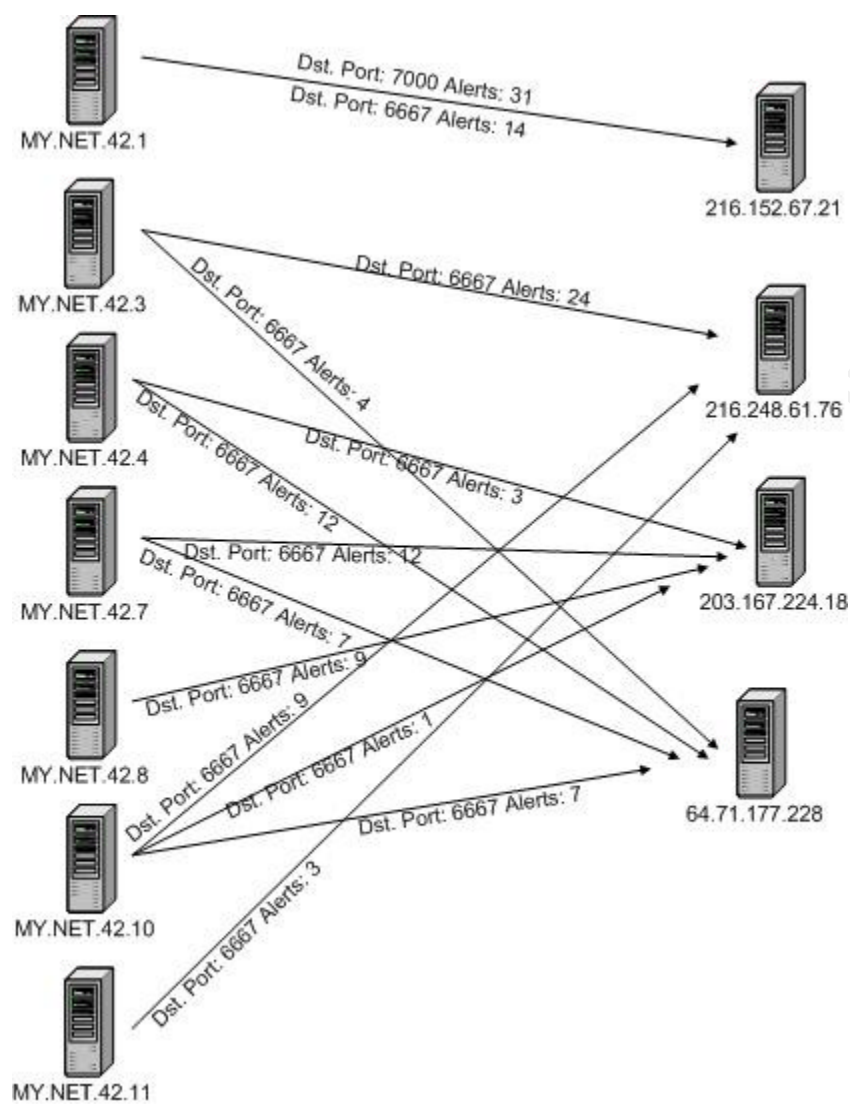
http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc

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

© SANS Institute 2004. All rights reserved. Author retains full rights.

APPENDIX: A Link Graph

The link graph below shows IRC Evil – Running XDCC activity on Internal Host.



IP Information for Top Destination involved in IRC Evil – Running XDCC Activity.

IP Address:216.152.67.21

HostName:216.152.67.21

OrgName: WebMaster, Incorporated
OrgID: WBMR
Address: 1601 Civic Center Drive, Suite 101
City: Santa Clara
StateProv: CA
PostalCode: 95050
Country: US

NetRange: 216.152.64.0 - 216.152.79.255
CIDR: 216.152.64.0/20
NetName: WEBMASTER-BLK-1
NetHandle: NET-216-152-64-0-1
Parent: NET-216-0-0-0-0
NetType: Direct Allocation
NameServer: NS1.WEBMASTER.COM
NameServer: NS1.WEBCHAT.ORG
Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate: 2000-07-18
Updated: 2003-09-05

TechHandle: MO21-ARIN
TechName: Owen, Mark
TechPhone: +1-408-345-1800
TechEmail: mark@webmaster.com

IP Address:216.248.61.76

HostName:216.248.61.76

CustName: Excelsior Media Studios
Address: Private Residence
City: Lexington
StateProv: KY
PostalCode: 40511
Country: US
RegDate: 2003-11-11
Updated: 2003-11-11

NetRange: 216.248.61.64 - 216.248.61.127
CIDR: 216.248.61.64/26
NetName: ABS-LXTNKY-EMS
NetHandle: NET-216-248-61-64-1
Parent: NET-216-248-0-0-1
NetType: Reassigned
Comment:
RegDate: 2003-11-11
Updated: 2003-11-11

TechHandle: VJN-ARIN
TechName: Newton, Valerie Jo
TechPhone: +1-814-260-2764
TechEmail: valerie.newton@telcove.com

OrgAbuseHandle: ABUSE167-ARIN
OrgAbuseName: Abuse
OrgAbusePhone: +1-814-260-2633
OrgAbuseEmail: abuse@adelphiabusiness.net

OrgTechHandle: AMB37-ARIN
OrgTechName: Barentine, Angela M
OrgTechPhone: +1-814-260-2757
OrgTechEmail: angela.barentine@telcove.com

OrgTechHandle: RSI4-ARIN
OrgTechName: Sirghie, Razvan
OrgTechPhone: +1-814-260-2756
OrgTechEmail: razvan.sirghie@telcove.com

IP Address:203.167.224.18

HostName:neptune.gameplanet.net.nz

inetnum: 203.167.224.0 - 203.167.224.63

netname: CLIX-CLIXACLD-NZ

descr: CLIX Auckland Red Network

country: NZ

admin-c: CCNO1-AP

tech-c: CCNO1-AP

remarks: Delegated by CLEAR Communications Ltd

remarks: 24/7 CLEAR NOC phone +64 9 912-4990

notify: netobjs@clear.net.nz

mnt-by: MAINT-CLIX-NZ

changed: netobjs@clear.net.nz 20010624

status: ASSIGNED NON-PORTABLE

source: APNIC

changed: hm-changed@apnic.net 20020827

role: CLEAR Communications Network Objects Maintainer

address: ISP Duty Officer, CLEAR Net Operations

address: CLEAR Communications Limited

address: Private Bag 92143

address: Auckland

country: NZ

phone: +64 9 912-5024

fax-no: +64 9 912-5008

e-mail: netobjs@clear.net.nz

admin-c: CCNO1-AP

tech-c: CCNO1-AP

nic-hdl: CCNO1-AP

remarks: For network abuse contact abuse@clear.net.nz

remarks: For 24/7 after-hours NOC, please call +64 9 912-4990

notify: netobjs@clear.net.nz

mnt-by: MAINT-CLIX-NZ

changed: netobjs@clear.net.nz 20010821

source: APNIC

IP Address:64.71.177.228

HostName:ca.enterthegame.com

ustName: Jeff Walter
Address: 42657 Newport Drive
City: Fremont
StateProv: CA
PostalCode: 94538
Country: US
RegDate: 2003-12-09
Updated: 2003-12-09

NetRange: 64.71.177.224 - 64.71.177.239
CIDR: 64.71.177.224/28
NetName: HURRICANE-CE1347-3B1
NetHandle: NET-64-71-177-224-1
Parent: NET-64-71-128-0-1
NetType: Reassigned
Comment:
RegDate: 2003-12-09
Updated: 2003-12-09

TechHandle: ZH17-ARIN
TechName: Hurricane Electric
TechPhone: +1-510-580-4100
TechEmail: hostmaster@he.net

OrgTechHandle: ZH17-ARIN
OrgTechName: Hurricane Electric
OrgTechPhone: +1-510-580-4100
OrgTechEmail: hostmaster@he.net

APPENDIX: C IP INFO. FOR TOP SOURCE HOST IN ALERT LOGS

IP Address: 220.12.4.217

HostName: YahooBB220012004217.bbtec.net

inetnum: 220.0.0.0 - 220.63.255.255
netname: BBTECH
descr: Japan nation-wide Network of SOFTBANK BB CORP
descr: Tokyo, Japan
country: JP
admin-c: SA127-AP
tech-c: SA127-AP
mnt-by: APNIC-HM
mnt-lower: MAINT-JP-BBTECH
changed: hostmaster@apnic.net 20020412
changed: hm-changed@apnic.net 20030616
status: ALLOCATED PORTABLE
source: APNIC

role: SoftbankBB ABUSE
address: 24-1, Nihonbashi Hakozaeki-Cho ,Chuo-Ku ,Tokyo
country: JP
phone: +81-0570-919-820

e-mail: hostmaster@bbtec.net
trouble: Please send spam report,virus alert
trouble: or any other abuse report
trouble: to abuse@bbtec.net
trouble: Any other Information, Notice,
trouble: Please send to hostmaster@bbtec.net
admin-c: TT123-AP
tech-c: ST222-AP
nic-hdl: SA127-AP
notify: admin@bbtec.net
mnt-by: MAINT-JP-BBTECH
changed: stsuruma@softbank.co.jp 20030613
source: APNIC

IP Address: 68.33.138.193

HostName: esx136dhcp705.essex01.md.comcast.net

Comcast Cable Communications, Inc. JUMPSTART-1 (NET-68-32-0-0-1)

68.32.0.0 - 68.63.255.255

Comcast Cable Communications, Inc. BALTIMORE-B-1 (NET-68-33-0-0-1)

68.33.0.0 - 68.33.255.255

IP Address: 68.32.127.158

HostName: pcp01823879pcs.howard01.md.comcast.net

Comcast Cable Communications, Inc. JUMPSTART-1 (NET-68-32-0-0-1)

68.32.0.0 - 68.63.255.255

Comcast Cable Communications, Inc. BALTIMORE-B-1 (NET-68-33-0-0-1)

68.33.0.0 - 68.33.255.255

© SANS Institute 2004. Author retains full rights.

APPENDIX: B Scan Analysis

TOP SOURCE PORTS:

Source port	Hits ?	Hits bar
32783	2,653,396	
32788	440,311	
3300	371,058	
2279	219,052	
3521	157,730	
41446	129,274	
8767	97,246	
12203	87,165	
22321	67,754	
8888	62,925	
54811 other items	3,625,325	
Total	7,911,236	

TCP Port Scan Analysis:

Top Destination Hosts:

Count	Destination
20317	130.85.97.55
18221	61.156.14.122
8767	218.15.192.166
7967	130.85.112.222
6805	129.132.227.151
6718	216.188.76.37
6552	213.138.148.10
6048	198.247.172.10
6015	219.153.1.212
5160	195.41.53.186
4598	127.0.1.50
4393	81.94.131.69
4251	194.106.143.68
4250	64.224.20.142
4146	209.133.28.19
4117	130.85.12.6
4094	129.97.56.22
4027	202.103.30.41

UDP Scan Analysis:

Count	Top UDP Source
2793253	130.85.1.3
460558	130.85.1.4
370430	130.85.153.37
221628	130.85.111.34
157096	130.85.110.72
142556	130.85.110.72
73075	130.85.82.15
68290	130.85.70.207
65281	130.85.84.131
40883	130.85.72.155
33563	130.85.97.37
28753	130.85.153.97
26383	130.85.69.214
22084	130.85.153.87
18765	130.85.53.225
12216	130.85.42.7
11893	130.85.97.58
11684	130.85.69.226
10417	130.85.97.62
7078	130.85.97.40

Count	Top Destination
83490	69.6.33.10
75513	69.6.33.11
64808	192.26.92.30
38437	203.20.52.5
38192	192.5.6.30
35335	192.55.83.30
30943	192.48.79.30
30622	131.118.254.34
29134	216.109.116.17
27868	131.118.254.33
27739	69.20.36.152
26798	69.20.36.154
25883	165.230.209.227
23554	69.6.33.9
23269	67.68.12.4
23121	209.92.188.201
22505	68.83.32.174
19851	128.194.254.5
19849	69.6.27.88
19689	131.118.254.35

Some Interesting Probes in Port scan Logs:

Count	Top Source	Dst. port	Description
11597	211.110.127.133	4899	?Radmin
7618	61.107.181.148	4899	?Radmin
15192	69.0.147.187	4000	icq
5360	80.222.6.95	4000	icq
10059	61.155.11.30	554	Real server
11177	212.27.197.78	5900	vnc
8180	216.99.99.120	555	?Raman Worm
8001	172.210.122.200	7300	?Kuang2

© SANS Institute 2004, Author retains full rights.