# Global Information Assurance Certification Paper

# Intrusion Detection:
# New Tools and Existing Theory



***Marcus Wu***
***GCIA Practical version 3.3***
***Submitted: January 23, 2003***

# Assignment #1: Describe the State of Intrusion Detection

*Applying Support Vector Machines to Intrusion Detection*

## Introduction

Intrusion detection is a time consuming process. It can be so time consuming that it cannot be done without the help of software. So far, programs have been developed to detect intrusions based on patterns and signatures or by looking for deviations from the normal pattern of network traffic and system operations. These programs are commonly known as intrusion detection systems.

Unfortunately, current intrusion detection systems are not perfect. As intrusion detection systems find anomalies, they report them as events. The events reported by intrusion detection systems may not truly signify hostile action; these events are known as false positives. Intrusion detection is known to be subject to many false positives because anomolous traffic often looks very similar to normal, authorized traffic.

As more research is done in the area of IDS, intellegent algorithms are being created to eliminate many false positives. In "Support Vector Machines - Background and practice," Panu Erasto states "In many fields of science, computer science in particular, automatic learning from examples is a long-standing goal." An algorithm that could learn from example would prove to be a formidable intrusion detection system. One such algorithm which attempts to learn from example is called support vector machines or SVM.

## SVM: A definition

The support vector machine is a classification and regression algorithm. Classifying data into groups is a very difficult task to topple. Support vector machines attempt to classify data points into groups by determining what side of a hyperplane a point of data exists on. A hyperplane is similar to a two dimensional plane, except that it exists in more than three dimensions. As a SVM is trained, it determines optimal hyperplanes which can separate the data points into the classes or categories it is told that they belong to. After the SVM is trained, it still learns from additional datapoints by altering or moving the hyperplanes that it already has. Thus, SVMs can cope with data for which it has never seen and can improve on its own accuracy.

## How SVMs work

Support vector machines are trained on a set of data points for which the trainer knows the correct classification. The data set is comprised of data points with a number of attributes. For instance, if we were classifying areas of three dimensional space, the attributes we would use are the x, y, and z coordinates. Each attribute that comprises a data point is called a dimension; thus, in the previous example the axis x,y, and z would be the three dimensions of each data point. The SVM would use those attributes to determine a hyperplane that could separate the data. A very simple example could be that a room is divided into two parts. Random coordinates from the room were recorded along with what side of the room those coordinates were in. After training the SVM, it would be able to tell you what side of the room any given point were in. This sounds very simple, but immagine a room that

existed in twelve dimensional space which was divided into 100 parts: an SVM could still classify points in the room to its 100 sections, given that it was trained well. The previous example of a room in three dimensional space was very simplistic, since classification was based on sides of a room, the separation between classes is obvious. The support vector machine algorithm can handle cases where classification is not on a clear linear plane. To limit the scope of this document, details on how the support vector machine algorithm handles linearly inseparable cases will not be discussed.

## Support Vector Machines as an IDS

The task of deciding whether data received over a network constitutes an attempt at information gathering or an attempt at compromising security, whether successful or not, is a difficult task. Even an experienced intrusion analyst may mis-categorize an event if a detail is missed. If a program is to be used to assist the intrusion analyst, it must be very accurate, and it must be very fast in order for it to handle the volume of a busy network. Support vector machines can meet both of those requirements. In "Suport Vector Machines -- Backgrounds and Practice, Panu Erasto describes the increasing need to use learning algorithms, "Also, in recent years the amount of information that has to be processed has exploded and there is a growing need to extract structure from the data instead of just storing it." Erasto goes on to say, "Morevoer, if one is able to capture some dependence in the data this knowledge can be used to predict future situations."

Using support vector machines would decrease the amount of traffic an intrusion analyst would be required to review. SVMs would give a classification as to the type of event and reduce the number of false positives seen by traditional IDSs. Support vector machines would give more time to the intrusion analyst to understand an attack and determine its source; it therefore has the potential to prevent attacks from having as harsh of a consequence. Intrusion analysts with more time to respond to malicious traffic would be able to put more effort into listing their attackers at dshield or using their time to research the history of their attackers and the strategy of an attack. As networks get faster and larger, support vector machines grow in appeal as a way to reduce the number of false positives an intrusion analysist would have to look at in order to free their hands for more important parts of their occupation, namely research. Staying on top of intrusion detection is all about understanding both old and new vulnerabilities and attacks. Learning algorithms such as SVMs may give intrusion analysists the time they need to understand old attacks while researching newer ones.

## The Speed of an SVM

As more consideration is given towards using support vector machines in a production environment, one question that gains importance is, "How fast are support vector machines?" In "Intrusion Detection Using Support Vector Machines", while giving reasons to experiment with SVMs in intrusion

detection, the author states, "...as real time performance is of primary importance to intrusion detection systems, any classifier that can potentially outrun neural networks is worth considering." The algorithm for support vector machines was developed to handle massive amounts of data in reasonable amounts of time. It can handle such a large amount of data that researchers in the field of Bioinformatics are using it to classify large amounts of microarray gene expression data. Intrusion detection systems also see quite a bit of data on a single network. Support vector machines have proven their ability to process streams of data in the field of robotics where they were used to recognize objects within live video. On another note, support vector machines would not necessarily need to look at all of the data being sent and recieved on a network. A current IDS like snort could be used to detect anomolous traffic while the SVM could be used to further filter and classify the events in order to make the job of the intrusion analyst easier and faster. While this would lower the SVM's ability to encounter new information in order to learn new attack patterns, it would still be able to learn new signatures installed to the IDS in order to eliminate false positives.

## The Accuracy of an SVM

Accuracy of learning algorithms has always been scrutinized, especially in areas such as intrusion detection. It is not a trivial thing to trust a learning algorithm to something as important as network security. Much would have to be known about the accuracy of SVMs before they could be used in production within an intrusion detection solution. Support vector machines have been known to reach very high accuracies if they were trained well, being as close as within ten percent of being perfect. In a paper by Sriniva Mukkamala, he stated, "The testing set consisting of 6980 data points with 41 features, received 99.50% accuracy, with a total runtime of 1.63 sec." The following graph is from Mukkamala's paper and illustrates his results.

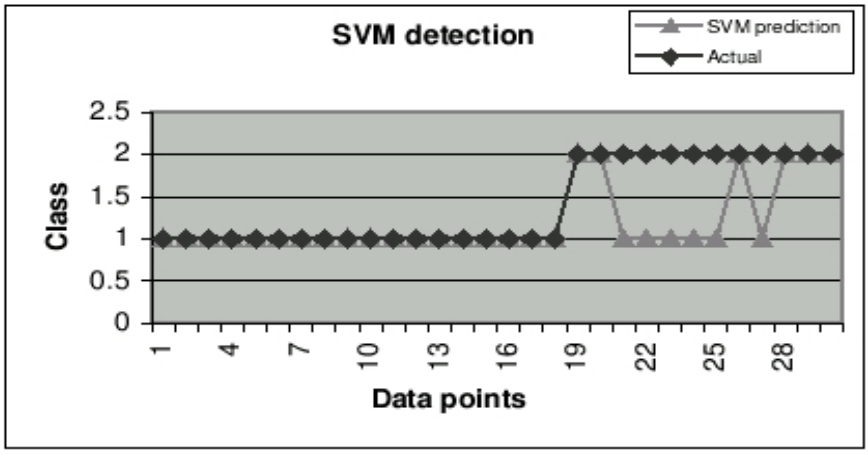



Fig. 1. SVMs results on KDD intrusion detection (outputs 1 denote 1 or normal; outputs 2 denote –1 or attack)

SVMs can adjust their hyperplanes as they come into contact with new data in orderto make themselves more accurate as time wears on. For the field of intrusion detection, one could train an SVM to bias classification towards being hostile to compensate for its slightly imperfect accuracy by making it alert more easily. These facts about the SVM algorithm have lead to its use in network intrusion detection systems. University research shows that the accuracy of SVMs reach and surpass that of neural networks.

## The Future of Learning Algorithms in Intrusion Detection

As networks grow faster and more information is transferred over networks, Support vector machines and other learning algorithms will gain a stronger foothold in intrusion detection. I do not believe that any learning machine will ever completely handle intrusions for a network because as fast and accurate as they are, people like to have the decision responsibility in what ultimately is decided as hostile and what actions are taken to protect themselves. Also, as the methods of detecting any attack get more complicated, it has been seen that the attacks themselves become more complicated and difficult to detect. Learning algorithm technology has not been in practice long enough for any prediction as to how easily it may be tricked. While it is accurate at recognizing objects in live video or recognizing genes within DNA, Those sets of data are not engineered by anyone in such a way as to try to trick the algorithm. It is almost guaranteed that information gathering and attack techniques will be developed to attempt to get past learning algorithms.

## Implementations of the SVM algorithm

While the support vector machine algorithm is relatively new to many fields including intrusion detection, I have found that there are many existing implementations of the algorithm. Here are two open source versions available for Unix/Linux.
libsvm - http://www.csie.ntu.edu.tw/~cjlin/libsvm/
svmlight - http://www.cs.cornell.edu/People/tj/svm_light/

libsvm includes Java source, so it can run on any system that has a Java runtime environment. The downloadable archive also contains C++ source and Windows binaries.

svmlight was developed on Solaris 2.5 with gcc. It also compiles on SunOS 3.1.4, Solaris 2.7, Linux, IRIX, Windows NT, and Powermac. Svmlight is free for scientific use.

## References

Erasto, Panu. "Support Vector Machines - Backgrounds and Practice." 2001. URL: http://ethesis.helsinki.fi/julkaisut/mat/rolfn/lt/erasto/supportv.pdf.

Janoski, Mukkamala, and Sung. "Intrusion Detection Using Neural Networks and Support Vector Machines." May 2002. URL: http://www.cs.nmt.edu/~IT/papers/hawaii7.pdf.

Janoski, Mukkamala, and Sung. "Intrusion Detection Using Support Vector Machines." 2002. URL: http://www.cs.nmt.edu/~IT/papers/hpccsandiagofinal.pdf.

Brown, Michael et al. "Knowledge-based Analysis of Microarray Gene Expression Data Using Support Vector Machines." January 4, 2000. URL: http://www.cse.ucsc.edu/research/compbio/genex/genex.html.

Chang and Lin. "LIBSVM -- A Library for Support Vector Machines." URL: http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Thorsten, Joachims. "SVMlight Support Vector Machine." URL: http://www.cs.cornell.edu/People/tj/svm_light/.

## Assignment #2: Three Network Detects

Three network detects are analyzed in this assignment. The first two were captured in real-time by an IDS. The first two were captured during an actual web server attack. The last detect was obtained from the incidents.org tcpdump binary logs.

### *Detect #1 Frontpage author.exe*

#### Source of Trace

This trace was taken from a snort device that picked up on Frontpage author.exe accesses to a corporate web server. The layout of the network basically consists of a DMZ where the web server is located and a Snort IDS in place.

## Detect Was Generated By

The snort alerts for this detect:

```
[**] [1:952:5] WEB-FRONTPAGE author.exe access [**]
[Classification: access to a potentially vulnerable web application]
[Priority: 2]
11/27-10:17:50.569847 200.148.107.128:1171 -> xxx.xx.81.10:80
TCP TTL:110 TOS:0x0 ID:58130 IpLen:20 DgmLen:428 DF
***AP*** Seq: 0x1DC02A  Ack: 0x1EF54BE0  Win: 0x5AC  TcpLen: 20

POST /_vti_bin/_vti_aut/author.exe HTTP/1.1..Date: Wed, 27 Nov 2
002 15:28:41 GMT..MIME-Version: 1.0..User-Agent: MSFrontPage/4.0
..Host: www.xxxxxxxxx.com..Accept: auth/sicily..Content-Length:
58..Content-Type: application/x-www-form-urlencoded..X-Vermeer-C
ontent-Type: application/x-www-form-urlencoded..Connection: Keep
-Alive....method=open+service%3a4%2e0%2e2%2e2611&service%5fname=
%2f.

[**] [1:952:5] WEB-FRONTPAGE author.exe access [**]
[Classification: access to a potentially vulnerable web application]
[Priority: 2]
11/27-10:45:04.226604 200.211.14.23:62333 -> xxx.xx.81.10:80
TCP TTL:112 TOS:0x0 ID:46436 IpLen:20 DgmLen:428 DF
***AP*** Seq: 0x13EC8E4  Ack: 0x360C379D  Win: 0x5B4  TcpLen: 20
```

```
POST /_vti_bin/_vti_aut/author.exe HTTP/1.1..Date: Wed, 27 Nov 2
002 13:51:30 GMT..MIME-Version: 1.0..User-Agent: MSFrontPage/4.0
..Host: www.xxxxxxx.com..Accept: auth/sicily..Content-Length:
58..Content-Type: application/x-www-form-urlencoded..X-Vermeer-C
ontent-Type: application/x-www-form-urlencoded..Connection: Keep
-Alive....method=open+service%3a4%2e0%2e2%2e2611&service%5fname=
%2f.

[**] [1:952:5] WEB-FRONTPAGE author.exe access [**]
[Classification: access to a potentially vulnerable web application]
[Priority: 2]
11/27-13:21:50.999167 200.158.156.222:1237 -> xxx.xx.81.10:80
TCP TTL:110 TOS:0x0 ID:3021 IpLen:20 DgmLen:453 DF
***AP*** Seq: 0x12827466  Ack: 0xB978E901  Win: 0x5AC  TcpLen: 20

POST /_vti_bin/_vti_aut/author.exe HTTP/1.1..Date: Wed, 27 Nov 2
002 18:29:42 GMT..MIME-Version: 1.0..User-Agent: MSFrontPage/4.0
..Host: www.xxxxxxxx.com..Accept: auth/sicily..Content-Length:
58..Content-Type: application/x-www-form-urlencoded..X-Vermeer-C
ontent-Type: application/x-www-form-urlencoded..Connection: Keep
-Alive..Cache-Control: no-cache....method=open+service%3a4%2e0%2
e2%2e3717&service%5fname=%2f.
```

*Figure 2.1.1 - Snort alerts for Frontpage Author.exe accesses*

It looks like our attacker has found a web server running Microsoft Frontpage 4.0 extensions. The attacker has attempted to use a known vulnerability in a possibly misconfigured host which would allow overwriting of files in the web root through posting to author.exe.

The Snort signature which generated this alert is below:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB -
FRONTPAGE author.exe access";flags:A+; urico ntent:"/_vti_bin/_vti_aut/author.exe"; nocase;
classtype:web-application-activity; sid:952;  rev:5;)
```

This signature alerts when /_vti_bin/_vti_aut/author.exe appears within a request sent to a web server on a web port.

## Probability Source Address was Spoofed

The probability that the source address was spoofed in this attack is low. HTTP requests are sent over an established TCP connection which would mean that the attacker had to complete the three-way handshake before making its request.

## Description of the Attack

author.exe has a vulnerability that, if not configured correctly would allow a remote user to connect to the web server's FrontPage extensions and gain full

access to the web server's web root directory. This would give the attacker the necessary access to accomplish web page defacement. Part of the full description of this attack (http://www.xato.net/Reference/webfolders.txt) describes using FrontPage webfolders:

"Essentially when you add a new WebFolder, Explorer wil l send a Post request to /_vti_bin/_vti_aut/author.dll (among others), which is installed as a part of the Fr ontPage Server extensions. So when you are using WebFolders, you are really just usin g the FrontPage Server extensions. If as an anonymous user you do not have read and execute access to that f ile, the server try to get an NTLM or Basic authenti cation from you. If any of those credentials succeed, you will now have a new WebFolder mapped to the remote server's web root."

## Attack Mech anism

There were several accesses to author.exe from three different sources. The first two sources were probably just scanning for vulnerable web servers. The last source actually exploited the vulnerability and defaced the web page. All three of the sources were determined to be Brazilian in origin. Although the accesses were from different sources, the effort was probably coordinated between them. The three sources may have all been compromised hosts operated by the same attacker.

## Correlations

Since the author.exe accesses came from different sources, any informational scans done previous to the attack have a higher probability of coming from an additional source owned by the same attacker or group of attackers. It would be difficult to pinpoint a specific probe or scan to this attack if it did come from a different source.

## Evidence of Active Targeting

This attack was targeted towards that specific web server. It was probably subject to a broader scan which highlighted the web server as a potential victim. After the system was determined to be vulnerable, multiple addresses were used to attack it to help conceal the identity of the attacker and to try to confuse anyone attempting to determine the cause of the defacement.

## Severity

The following formula is used to calculate severity:

*(Target's Criticality + Lethality of Attack) -(System Defense + Network Defense)*

| | | |
|---|---|---|
| *Criticality* | This attack was targeted at a web server. It is not as important to the network as a firewall or similar device, but is an important part of a corporate network. | *3* |
| *Lethtality* | The system's web root has been compromised. This allows | 4 |

| | | |
|---|---|---|
| *Criticality* | This attack was targeted at a web server. It is not as important to the network as a firewall or similar device, but is an important part of a corporate network. | *3* |
| | for the attacker to deface the main site. It also allows for executables to be sent and for his own custom ASP pages to be executed. | |
| *System* | A correct configuration of the FrontPage extensions would easily solve this problem, unfortunately this was not the case | 1 |
| *Network* | While this traffic cannot be blocked by a firewall since the server is a production web server, IDS was in place which detected the attack. | 3 |

When these values are plugged into our severity formula, we get a result of 4:
Severity = ( 3 + 4 ) - (1 + 3 ) = 3

## Defensive Recommendation

The best countermeasure for this type of attack is just to not use FrontPage web server extensions. If they are absolutely required, be sure to configure the file ACLs to highly restrict access to its modules. An article from Microsoft listed in the Resources for this detect describes how to set up the ACLs. One note is to use NTFS as the file system as opposed to FAT. FAT does not support full ACLs. If the system has already been compromised, no logs are created for the actions of the attacker. The options given to the attacker for compromising the host are so broad that a compromised host may have trojans installed that would be very difficult to detect. I would recommend a complete reinstall of the operating system for any system that is compromised with this particular exploit.

## Multiple Choice Question

Which of the following is not true about most windows based web servers?

A) Front page has executable access to many system dlls
B) Attacking a production web server through FrontPage will not be stopped by a firewall.
C) The permissions of the web author are usually greater than those given to IUSR_MACHINE.
D) A default FrontPage setup will protect everyone from author.dll based attacks.

The correct answer is D. Default FrontPage setups are vulnerable to author.dll based attacks. A, B, and C are all true statements on most Windows based web servers.

## References

Sozni. "webfolders.txt" -- a description of author.exe vunerability. URL: http://www.xato.net/Reference/webfolders.txt.

Microsoft. “Frontpage Security on IIS Systems.” URL: http://www.microsoft.com/technet/archive/office/office97/reskit/fp98serk/SECURITY.asp.

*Detect #2 /etc/passwd access attempt*

## Source of Trace

Here is the Snort trace for this detect:

```
[**] [1:1122:4] WEB-MISC /etc/passwd [**]
[Classification: Attempted Information Leak] [Priority: 2]
01/19-07:40:06.982612 200.34.237.254:2601  -> xxx.xxx.73.8:80
TCP TTL:116 TOS:0x0 ID:6547 IpLen:20 DgmLen:177 DF
***AP*** Seq: 0xC22E3CF  Ack: 0x527A5768  Win: 0x2238  TcpLen: 20

HEAD /../../../../../../../../../../etc/passwd?/c+dir+c:\ /%2
E%2E/%2E%2E/%2E%2E/etc/passwd

[**] [1:1002:5] WEB-IIS cmd.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
01/19-07:40:35.348475 200.34.237.254:2638  -> xxx.xxx.73.8:80
TCP TTL:116 TOS:0x0 ID:38804 IpLen:20 DgmLen:214 DF
***AP*** Seq: 0xC22E599  Ack: 0x52F76B48  Win: 0x2238  TcpLen: 20

HEAD /_mem_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe?/c
+dir+c:\ r+c:\ HTTP/1.0..Host: xxx.xxx.73.8..Content-Type: text/
html; charset=UTF-8..Content-Length: 164
```

The first trace matched Snort signatures for a /etc/passwd retrieval attempt. The second trace was matched as a cmd.exe access shortly after the passwd retrieval attempt. No other activity from the attacker matched a Snort signature or was recorded by the web server. These attacks were directoed towards a web server on a DMZ with Snort IDS.

## Detect Was Generated By

The Snort signatures which matched the events look for traffic to a web server on web server ports which contain the content of /etc/passwd or cmd.exe respectively.

Many well known exploits and vulnerabilities rely on accesses to these two files, however they are not often considered related to each other.

The Snort signatures which matched these events are below:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-MISC
/etc/passwd"; flags:A+; content:"/etc/ passwd"; nocase; classtype:attempted-recon; sid:1122;
rev:4;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS
cmd.exe access"; flags:A+; content:"cmd.exe"; nocase; classtype:web-application-attack;
```

sid:1002; rev:5;)

## Probability Source Address was Spoofed

Both signatures that were matched indicate that an http connection was made. In both cases, a full TCP three way handshake had to have been completed making it very unlikely that the source address had been spoofed.

## Description of the Attack

Both accesses to /etc/passwd as well as to cmd.exe require escaping the http server's web root directory. Most web servers will prevent this type of activity, however it has been discovered that some http servers are vulnerable to be tricked into escaping the web root directory if URI encoding is used.

## Attack Mechanism

The web server may accept these URLs depending on what order they are decoded and checked for path traversals. Consider the following:

%2E translates to '.'
/%2E%2E/%2E%2E/%2E%2E/etc/passwd tra nslates to /../../../etc/passwd

While both of those are equivalent URLs, the order in which they are authorized and decoded could allow the URL to be processed. For instance, if the URL is checked for path traversals before it is decoded, the URL will pass through the URL check. If the URL is decoded first, when it is checked for path traversals it will not pass through the URL check.

An attack using this sort of URI encoding trick entirely depends on the http server checking for path traversals before decoding the URL.

Looking further into the attacks, I noticed that /etc/passwd was passed parameters. The parameters translate out to /c dir c:\. This is something you would expect to see after an attempt to access cmd.exe to try to list the contents of the root directory. Appending those parameters to a /etc/passwd access seems rather pointless. This leads me to believe that this attack was scripted and configured improperly.

## Correlations

A Google search resulted in several matches to the path traversal type of attack. A few pages worth visiting are:

http://www.iss.net/security_center/advice/Intrusions/2000645/default.htm
http://cert.uni-stuttgart.de/archive/intrusions/2002/11/msg00004.html
http://archives.neohapsis.com/archives/snort/2002-10/0875.html

The last of those is an archive of the incidents.org mailing list in which the the poster states his opinion that the /etc/passwd access is the result of a misconfigured script. This reinforces my ideas about the attack.

## Evidence of Active Targeting

It seems that in our case, the attacker does not know whether the server is running a Unix system or a Windows system. The attack may have been part of a broad scan for vulnerable web servers. If the attack had been directed, a simple http header grab or an attempt to retrieve an inexistent document could have easily been used to return information regarding the type of server as well as the version. Since the attack attempted to retrieve a Unix password file as well as access a windows executable, it would be safe to assume that the attacker knows very little about the web server.

An attacker attempting a directed attack would most likely research their target and only attempt those attacks for which the victim would most likely be vulnerable. This would save the attacker's time as well as preventing extraneous IDS alerts. The alerts observed here were quite the opposite: the web server was not vulnerable to either attack, and the the attacker obviously did not even know what operating system the web server was running on.

Furthermore, the /etc/passwd access was given parameters as if the attacker were trying to execute it in a similar fasion to cmd.exe. Not only does the attack look undirected, but it also looks clumsy and foolish.

## Severity

The following formula is used to calculate severity:

*(Target's Criticality + Lethality of Attack) -(System Defense + Network Defense)*

| | | |
|---|---|---|
| *Criticality* | This attack was targeted at a web server. It is not as important to the network as a firewall or similar device, but is an important part of a corporate network. | *3* |
| *Lethtality* | The system is not vulnerable to this attack. Either the current build is not vulnerable or the software has been patched against the particular exploit or attack. | 1 |
| *System* | Network IDS was in place and detected the activity, and the http server software was not vulnerable to the attack. | 4 |
| *Network* | The traffic observed cannot be blocked by a firewall since it is on a valid port destined for a valid web server. | 1 |

When these values are plugged into our severity formula, we get a result of 4:
Severity = ( 3 + 1 ) - (4 + 1 ) = -1

## Defensive Recommendation

Double check to make sure that any web servers on your network are the latest versions available and that they have all applicable patches applied. Microsoft security bulletin MS00-078 describes the directory traversal vulnerability and provides patches: http://www.microsoft.com/technet/security/bulletin/ms00-078.asp

Testing can be done to check for vulnerability to path traversal techniques to the web server by sending those requests to your web server manually. Many available web server vulnerability scanners will check for susceptibility to such path traversals.

## Multiple Choice Question

%E0%81%9C is the URI encoding for which common ASCII character?
A) \
B) /
C) ?
D) ;

The correct answer is A. %E0%81%9C is the three byte encoding for \.

*Detect #3 IRC Nick Changes*

## Source of Trace

First of all, here are the offending snort alerts:

```
[**] [1:542:8] CHAT IRC nick change [**]
[Classification: Misc activity] [Priority: 3]
11/14-15:59:25.476507 170.129.50.120:61599 -> 217.8.139.18:6667
TCP TTL:123 TOS:0x0 ID:1429 IpLen:20 DgmLen:56 DF
***AP*** Seq: 0xADB829FB  Ack: 0xF76B24A6  Win: 0x3C30  TcpLen: 20

[**] [1:542:8] CHAT IRC nick change [**]
[Classification: Misc activity] [Priority: 3]
11/14-15:59:25.956507 170.129.50.120:61599 -> 217.8.139.18:6667
TCP TTL:123 TOS:0x0 ID:1479 IpLen:20 DgmLen:56 DF
***AP*** Seq: 0xADB82CA7  Ack: 0xF76B2A82  Win: 0x4038  TcpLen: 20

[snip]

[**] [1:542:8] CHAT IRC nick change [**]
[Classification: Misc activity] [Priority: 3]
11/14-16:00:41.446507 170.129.50.120:61626 -> 66.159.16.174:6667
TCP TTL:123 TOS:0x0 ID:9284 IpLen:20 DgmLen:56 DF
***AP*** Seq: 0xB8903557  Ack: 0xA6EF79E  Win: 0x3E93  TcpLen: 20

[**] [1:542:8] CHAT IRC nick change [**]
[Classification: Misc activity] [Priority: 3]
11/14-16:00:41.736507 170.129.50.120:61626 -> 66.159.16.174:6667
TCP TTL:123 TOS:0x0 ID:9320 IpLen:20 DgmLen:56 DF
***AP*** Seq: 0xB8903602  Ack: 0xA6EF90F  Win: 0x3D22  TcpLen: 20
```

These alerts were obtained from the tcpdump binary logs located at http://www.incidents.org/logs/Raw/. This detect was discovered in the file 2002.10.14 and related detects continue in 2002.10.17. The listings below are taken from the tcpdump binary logs that were produced by tcpdump running against both the 2002.10.14 and 2002.10.17 files.

```
2002.10.14

---------------

15:59:25.476507 170.129.50.120.61599 > 217.8.139.18.6667: P 2914527739:2914527755(16) ack
4150994086 win 15408 (DF)

0x0000   4500 0038 0595 4000 7b06 b916 aa81 3278        E..8..@.{.....2x

0x0010   d908 8b12 f09f 1a0b adb8 29fb f76b 24a6        ..........)..k$.

0x0020   5018 3c30 3523 0000 4e49 434b 2052 3030        P.<05#..NICK.R00

0x0030   7465 442d 3030 340a                            teD-004.

15:59:25.956507 170.129.50.120.61599 > 217.8.139.18.6667: P 684:700(16) ack 1501 win
16440 (DF)

0x0000   4500 0038 05c7 4000 7b06 b8e4 aa81 3278        E..8..@.{.....2x

0x0010   d908 8b12 f09f 1a0b adb8 2ca7 f76b 2a82        ..........,..k*.

0x0020   5018 4038 2893 0000 4e49 434b 2052 3030        P.@8(...NICK.R00

0x0030   7465 442d 3030 340a                            teD-004.

15:59:26.316507 170.129.50.120.61599 > 217.8.139.18.6667: P 1368:1384(16) ack 3001 win
16440 (DF)

0x0000   4500 0038 05f3 4000 7b06 b8b8 aa81 3278        E..8..@.{.....2x

0x0010   d908 8b12 f09f 1a0b adb8 2f53 f76b 305e        ........../S.k0^

0x0020   5018 4038 200b 0000 4e49 434b 2052 3030        P.@8....NICK.R00

0x0030   7465 442d 3030 340a                            teD-004.
```

*Figure 2.3.1 - Snort alerts from 2002.10.14*

```
2002.10.17

---------------

03:50:28.956507 170.129.50.120.65037 > 216.12.211.209.7000: P 4216235350:4216235371(21)
ack 2038236175 win 16116 (DF)

0x0000   4500 003d ce83 4000 7b06 a85f aa81 3278        E..=..@.{.._..2x

0x0010   d80c d3d1 fe0d 1b58 fb4e a556 797d 040f        .......X.N.Vy}..

0x0020   5018 3ef4 256e 0000 4e49 434b 205b 5356        P.>.%n..NICK.[SV

0x0030   4344 505d 2d58 4443 432d 3335 0a               CDP]-XDCC-35.

03:50:29.126507 170.129.50.120.65037 > 216.12.211.209.7000: P 73:94(21) ack 133 win 15984
(DF)

0x0000   4500 003d ce99 4000 7b06 a849 aa81 3278        E..=..@.{..I..2x

0x0010   d80c d3d1 fe0d 1b58 fb4e a59f 797d 0493        .......X.N..y}..

0x0020   5018 3e70 2525 0000 4e49 434b 205b 5356        P.>p%%..NICK.[SV

0x0030   4344 505d 2d58 4443 432d 3335 0a               CDP]-XDCC-35.
```

*Figure 2.3.2 - Snort alerts from 2002.10.17*

The previous listings were obtained by running:

```
$ /usr/sbin/tcpdump -X -n -r $file src host 170.129.50.120 and \(
dst port 6667 or dst port 7000 \)
```

These options look for traffic originating from 170.129.50.120 and destined for two common irc server ports.

## Detect Was Generated By

This detect was found using a Snort Intrusion Detection System version 1.9.0 build 209. The alerts were generated by running Snort with parameters to tell

it to read the tcpdump binary data and write the alerts to a mysql database. A modified snort.conf was used to allow snort to make use of signatures that are not commonly enabled. A simple script was used to run snort against all of the raw tcpdump format logs found at incidents.org. The script is listed in figure 2.3.3.

```
#!/bin/bash
for file in `ls 2002*`
do
  snort -o -d -c /home/riptide/giacdetectdata/rules/snort.conf -r $file
done
```

*Figure 2.3.3 - Script used against incidents.org tcpdump format logs*

After the alerts were stored into the database, a web page interface, ACID version 0.9.6b22, was used to view the alerts based on many different criteria.

## Probability Source Address was Spoofed

The packets analyzed in this detect are part of an established TCP connection. Chances that these packets are spoofed is highly unlikely. Although the initial TCP three way handshake was not captured by the snort ruleset that originally logged the data, we can be reasonably sure that the data was not spoofed.

## Description of the Attack

The alerts first caught my eye because they were part of a connection to an IRC server which is considered a haven for mischievous characters. The initial Snort alert is not very alarming, but as I looked down the list of the alerts in ACID, I noticed that there are quite a few name changes and they all are timestamped within a second of each other. Some of them being within the same second. Those name changes were abnormally fast, and probably signify a script being used, so I took a closer look at the alerts.

When actually viewing one of the alerts, I quickly noticed the nick that the user was requesting, "R00teD-004." Rooted is a term used by hackers that they use to refer to a system on which the administrator account has been compromised. While that name as an IRC nick does not mean that the host in concern has been compromised, it certainly is alarming. Another thing to note is the appearance of numbers at the end of the nick. The numbers make me feel as though the nick is a script-generated nick and that there may be a R00teD-001 through R00teD-003 and possibly -005 and on.

Looking across all of the alerts generated from the file 2002.10.14, the nick remains constant, but the name changes are very frequent in time. This leads me to believe that maybe the script chooses its name based on what nicks are available, and that it keeps checking for lower numbered nicks open in a brute force type method. The problem is that it's always requesting the same

nick. Possibly, the algorithm is not very smart and submits a nick change for its current nick if a new nick is not available. It is common for trojans and scripts to be programmed quickly with little regard for efficiency.

After the host has its nick escapade in 2002.10.14, it doesn't show up until the file 2002.10.17. At this point, the host connects back to IRC, but to a different server. This time it chooses "[SVCDP]-XDCC-35" as its nick. The XDCC in its name signifies that the client is an XDCC bot. Dslreports.com hosts a frequently asked questions on their server about Internet Relay Chat. The frequently asked questions states the following in regards to XDCC bots:

```
Q: What's an XDCC? (#4493)
A: With IRC in full swing, XDCC bots are common sights in channels these days. An XDCC
is a bot that has certain packets uploaded to it. These packets may be anything from the
recent game to a good movie. XDCCs are usually r00ted (hacked), and transfer at very high
speeds because they are on fast lines.
```

*http://www.dslreports.com/faq/4493*

Although no other packets are captured in the logs at incidents.org, I feel that it is safe to assume that the host in concern has a compromised account, possibly root, which is being used to connect to IRC as an XDCC bot.

## Attack Mechanism

It is unknown how the internal host was compromised to begin with. Some Gnutella connections were found to have been made prior to the host connecting to IRC. It is possible that a file received from Gnutella was a trojan. Once the system was connecting to IRC, it seems as though it was stopped, and reconfigured as an XDCC bot. This leads me to think that the original compromising attack was not a directed, human driven attack; if it were, the host would probably have been configured as the XDCC bot the first time. It is unknown what events took place during the reconfiguration of the bot. Whether other hacker utilities were installed at this time is unkown.

## Correlations

After looking for additional network traffic originating from the compromised host near the time of it being compromised, many Gnutella connections are seen. Perhaps a file retrieved from Gnutella is responsible for compromising the host.

I was unable to find any anomolous traffic between the time that the first set of IRC name changes occur and when the XDCC bot is started.

Very little was known about this particular compromise. The only thing I had to work from were the nick changes. It was difficult to find anything to correlate with. I did searches on Google for the irc nicks that I saw, but they were not specific enough to pick up anything that I could say for sure was the same irc script or trojan.

## Evidence of Active Targeting

From the information I gathered, it does not look as if the host was targeted until it was reconfigured as an XDCC bot. The original compromise was probably the result of a widely sent trojan which may have originated from one of the host's many Gnutella connections.

## Severity

The following formula is used to calculate severity:

*(Target's Criticality + Lethality of Attack) -(System Defense + Network Defense)*

| | | |
|---|---|---|
| *Criticality* | The type of host is most likely an end user system probably running a form of windows. | *1* |
| *Lethtality* | The system is compromised to an unknown extent. Access to a shell is probable. | 4 |
| *System* | No host based IDS or personal firewall appeared to be in place. | 1 |
| *Network* | None of the traffic observed seemed to be hindered by any defensive mechanisms in place | 1 |

When these values are plugged into our severity formula, we get a result of 4:
Severity = ( 1 + 4 ) - (1 + 1 ) = 3

## Defensive Recommendation

I would recommend blocking IRC traffic at the firewall as well as Gnutella and other peer-to-peer applications. These services are mostly unproductive in a business oriented network, and should not be used. A network usage policy should be in place to inform end users of these restrictions.

It is unknown whether virus scanning software is in use on the host and also unknown whether the trojan or XDCC bot would be detected by a virus scanner, but I would recommend a strong virus solution if one is not in place.

For some situations, you might configure Snort Flex Resp to send RSTs to both ends of the connection when certain signatures are matched. Unfortunately, the signature which matched in this circumstance was just a nick change. Closing the connection on IRC nick changes prevent most users from connecting to IRC and would almost be equivalent with blocking all IRC connections. Also, the IRC nick change signature is not a rule which is enabled by default. It would probably not be enabled on most networks since it would be the source of a lot of noise.

In a University situation, I would keep all sensitive computers within a more secure, trusted network. Any systems not under any central administration like computers within dorms should be kept on an untrusted network with a

firewall and IDS in place between them the trusted network.

While it might not be possible to block this sort of traffic to all systems, it can be prevented from having an adverse affect on parts of the network that can be controlled. Some users in the untrusted section of the network may still want some protection from this sort of thing, and for that I would suggest having information available to those users about choices of virus software and personal firewalls.

### Multiple Choice Question

What ports do Internet Relay Chat servers commonly Listen on?
A)6667-7000
B)6667
C)6660-7000
D)9999

The correct answer is C. A is too restrictive on the low range. B is just the most common single port. D is not a common port used to connect to IRC servers.

## Assignment #3: Analyze This

### *Executive Summary*

There is a large amount of data that needs to be analyzed in this part of the assignment. The tools chosen can either greatly increase the amount of hand work, or greatly decrease it. Since the Snort ruleset which generated these alerts is a mostly default ruleset, there will probably be a lot of noise produced in the alerts. The trick in intrusion detection is finding tools that can give the analyst a view of the data which will give insight into the trends in the alert data while also allowing easy ways to find information about individual alerts.

In this part of the practical, I used a few scripts to help me process and format the data in a way that it could be easily analyzed. I used a PHP script to parse the alerts files and insert to a Snort database to use with the ACID web page interface. I also used perl and shell scripts from Steven Drew practicals to parse scans and oos data.

### *Files Analyzed*

In this third assignment, five consecutive days of Snort logs were to be chosen to be analyzed. I chose the logs covering the period from January 15, 2003 to January 19, 2003. These logs were generated by an unknown version of Snort, and a ruleset which is described as "fairly standard" in the assignment description.

The alert files I chose for analysis are shown below:

| | |
|---|---|
| alert.030115 | 15541059 bytes |
| alert.030116 | 16325085 bytes |
| alert.030117 | 17887897 bytes |
| alert.030118 | 24985274 bytes |
| alert.030119 | 23281720 bytes |

The scans files I chose for analysis are as follows:

| | |
|---|---|
| scans.030115 | 33369113 bytes |
| scans.030116 | 35388075 bytes |
| scans.030117 | 43519917 bytes |
| scans.030118 | 97265545 bytes |
| scans.030119 | 91232263 bytes |

Finally, the out of spec files I chose are these:

| | |
|---|---|
| OOS_Report_2003_01_15_21827 | 798723 bytes |
| OOS_Report_2003_01_16_30391 | 696323 bytes |
| OOS_Report_2003_01_17_22332 | 890883 bytes |
| OOS_Report_2003_01_18_6261 | 348163 bytes |
| OOS_Report_2003_01_19_19130 | 317443 bytes |

### *Highest Occurring Detects*

These are the events which I found to be the most prevalent in the alert files. I also include the first and last occurance time of the events as well as how many unique source and destination addresses are observed triggering the specific signature. This information was easily gathered from the ACID web interface to the Snort database which I added all of the events from the alert files to. ACID had to be reconfigured slightly to show more signatures under the most frequent alerts report. I did this by modifying the acid_conf.php.

| # | Signature | Total # | # Src's | # Dest's | First | Last |
|---|---|---|---|---|---|---|
| 1 | High port 65535 tcp - possible Red Worm - traffic | 78606 (39%) | 192 | 187 | 2003-01-15 00:00:16 | 2003-01-19 23:46:02 |
| 2 | Watchlist 000220 IL-ISDNNET-990517 | 35073 (17%) | 72 | 91 | 2003-01-15 00:04:04 | 2003-01-19 23:45:15 |
| 3 | SMB Name Wildcard | 25708 (13%) | 768 | 925 | 2003-01-15 00:00:14 | 2003-01-19 23:30:15 |
| 4 | spp_http_decode: IIS Unicode attack detected | 22414 (11%) | 517 | 678 | 2003-01-15 00:30:59 | 2003-01-19 23:45:37 |
| 5 | TFTP - External UDP connection to internal tftp server | 21043 (10%) | 7 | 3 | 2003-01-15 00:07:10 | 2003-01-19 23:17:55 |
| 6 | High port 65535 udp - possible Red Worm - traffic | 3865 (2%) | 158 | 144 | 2003-01-15 00:03:49 | 2003-01-19 23:33:27 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 7 | Possible trojan server activity | 2529 (1%) | 28 | 47 | 2003-01-15 13:49:15 | 2003-01-19 19:19:51 |
| 8 | Incomplete Packet Fragments Discarded | 2390 (1%) | 19 | 17 | 2003-01-15 21:14:47 | 2003-01-19 16:40:01 |
| 9 | spp_http_decode: CGI Null Byte attack detected | 1897 (1%) | 47 | 80 | 2003-01-15 08:27:16 | 2003-01-19 22:54:38 |
| 10 | IDS552/web-iis_IIS ISAPI Overflow ida nosize | 1722 (1%) | 1494 | 645 | 2003-01-15 00:03:54 | 2003-01-19 19:02:13 |

You may notice that there is a dramatic decrease in the total number of alerts between the fifth and sixth most frequent alerts. That looks like a good place to draw the line. The top five most frequent alerts will be analyzed further.

## High port 65535 tcp - possible Red Worm - traffic

This signature matched 78606 times. That's a total of 39% of all alerts. There were 192 different source addresses and 187 unique destination addresses observed associated with this Snort signature.

Since I am not particularly familiar with this specific signature, I decided to look the Snort signature up, and see how it triggers. Unfortunately, I could not find this signature in my rules files, so I downloaded an earlier version of the rules. I could not find this signature in any of the rulesets I downloaded, so I dis some searching on Google, and discovered that the Red worm has been renamed to Adore worm.
I still could not find signatures for the Adore worm, but after more searching on Google, I discovered http://www.sans.org/y2k/adore.htm which gives a full description of the worm. Within this description, it states:
"Adore then runs a package called icmp. With the options provided with the tarball, it by default sets the port to listen too, and the packet length to watch for. When it sees this information it then sets a rootshell to allow connections."
Another document on the Adore worm located at
http://www.sans.org/rr/threats/mutation.php states that once the icmp program recieves a packet, it opens a backdoor on TCP to port 65535. This is the traffic that the signature that captured these events is looking for.
This worm is a pretty nasty one, so attention should be paid to these alerts.
These are all of the MY.NET IP addresses involved with these alerts:

| | | |
|---|---|---|
| MY.NET.84.151 | MY.NET.88.193 | MY.NET.104.204 |
| MY.NET.108.34 | MY.NET.198.220 | MY.NET.91.252 |
| MY.NET.114.88 | MY.NET.88.165 | MY.NET.150.215 |
| MY.NET.6.40 | MY.NET.87.7 | MY.NET.118.6 |
| MY.NET.84.193 | MY.NET.91.104 | MY.NET.113.4 |
| MY.NET.150.16 | MY.NET.29.3 | |

Many of these may be false positives because 65535 is a valid high port for the return port of outgoing traffic. These alerts could be falsely generated if that port was chosen for returning traffic and the firewall is stateless or looses state.

Although some of these may be false positives, MY.NET.84.151 also appears on the Top Ten Talkers list. I would recommend using the Adorefind script located at:
http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm to find out whether any of these hosts is truly infected with Adore(Red) worm.

## Watchlist 000220 IL-ISDNNET-990517

This signature matched 35073 times. That's a total of 17% of all alerts. There were 72 different source addresses and 91 unique destination addresses observed associated with this Snort signature.

Once again, this is another signature for which I know nothing about. Looking through all of the Snort rules I have turned up nothing for this signature, so once again I rely on Google. According to a quick Google search and Brian Coyle's GCIA practical, Watchlists alert to traffic coming or going to a certain IP range.Looking at the number of unique source and destination IP addresses that are involved with these events is very revealing. There are only 72 sources and 91 destinations. This information further supports that the rule that generated these events is looking for traffic to a specific range of IP addresses.

This could alert to any sort of traffic to the range within the signature, so I looked at what the majority of the traffic was based on the ports used. The most prevalent ports were 25, 80, 1214, and 6346. Ports 25 and 80 are well known and frequently used ports, but 1214 and 6346 are not well known. After looking up what services use ports 1214 and 6346, it became obvious why there was so much traffic alerting to this signature: they are used for common peer to peer file sharing software. Port 1214 is commonly used for Kazaa, and port 6346 is commonly used for Gnutella.

While peer to peer file sharing networks are not considered exploits themselves, many files transferred over them may contain trojans or viruses. In addition, these file sharing networks tend to be used to transfer files whose contents usually break copywrite and piracy laws. Not only are they most likely illegal, but they are also normally very large in size and would severely reduce available bandwidth. My detect #3 was probably a direct result of a user downloading software from Gnutella which was packaged with a trojan horse.

Peer to peer file sharing application usage should be kept to a minimum or stopped completely. If it can be blocked completely on your network, I recommend it. If it cannot be blocked, keep the accesses to file sharing networks on an untrusted segment of your network.

## SMB Name Wildcard

This signature matched 25708 times. That's a total of 13% of all alerts. There were 768 different source addresses and 925 unique destination addresses observed associated with this Snort signature.

This is another rule that I could not find in my current Snort ruleset. This one is a little easier to guess at, though. The rule symbolises an attempt to access SMB services from an external address into the network which Snort was running on. The Snort signature would probably look something like this:

```
alert udp any any -> $HOME_NET 137 (msg:"SMB Name Wildcard";
content:"CKAAAAAAAAAAAAAAAAAAAAAA AAAAAAA|0000|";)
```

A post on the Snort-users mailing list, http://archives.neohapsis.com/archives/snort/2000-08/0289.html explains that such requests are used to obtain the names of remote systems' in their netbios configuration. If the probe succeeds, the attacker can then mount shared drives on the system that responded.

Looking further into the traffic, I noticed that each source address probed about 60 or more times to different destinations within the MY.NET network. These alerts probably were generated from attackers who were searching for systems that they could abuse through netbios. These probes are informational in nature, but if any information is returned, it will most likely be used to compromise the filesystem of the systems which have been configured to allow any guests to access the filesystem. Even if write access is not permitted on any shares found, it may be possible to retrieve email, email server passwords, copywrited software, and any other information that the hacker may deem worthy. Shared filesystems are not the only available targets to a potential hacker. Many other devices and medias can be shared over netbios.

Tod Beardsley reported a high number of occurances of these alerts in his practical as well, but he dismisses it as normal NetBIOS name resolution on the internal network since most of his alerts were generated from the internal network. Unfortunately, this is not the case for us this time. I believe that between the time he wrote his practical and now, the alert had been modified to only alert to external addresses connecting to the internal network.

SMB attempts and scans to MY.NET from the outside and from so many different sources is rather daunting. I would ensure that this is not a problem by configuring a firewall or border router to drop any attempted traffic destined to port 137.

## spp_http_decode: IIS Unicode attack detected

This signature matched 22414 times. That's a total of 11% of all alerts. There

were 517 different source addresses and 678 unique destination addresses observed associated with this Snort signature.

This signature indicates that the http_decode preprocess of Snort didn't like some of the unicode representations of characters that it found within traffic to a web server. According to John Berkers on the Snort-users mailing list, "These events are sometimes triggered by visiting sites that use multi-byte characters such as Simplified Chinese etc."

After looking through the sources and destinations for many of the alerts, I did notice that many of them were destined to web servers external to MY.NET. Those accesses directed towards external web servers are likely to be triggering the preprocessor trigger because of character sets for languages like Chinese which use multiple bytes as John Berkers pointed out.

This preprocessor seems to be quite noisy, and consideration might be put towards disabling it to avoid distracting the analyst from true attacks. However, on the other side of the argument, the signature for this event is a rather important one if any IIS servers are running on MY.NET. Perhaps looking into tuning it to be a bit less noisy would be a better solution. As noisy as this signature is right now, even if a true attack were to occur which this signature would catch, it would be difficult to separate and diagnose it when it is mixed in with all of the other noise.

Further analyzing these events, I looked up the hostnames of many of the external web servers which were accessed from internally to generate some of the events in question. Many of the hostnames were hosts known to offer a lot of streaming media. Without additional information like raw network dumps of the traffic, it is difficult to say for sure, but I believe that streaming media from web pages may also cause this signature to trigger on a false positive.

There is still the traffic inbound to MY.NET web servers which triggered the signatures to consider. There are still the same causes of false positives that can be considered for the inbound traffic, however since the traffic is inbound, I would pay more attention to it. Perhaps the rule can be completely disabled if all inbound traffic for this signature was found to be false possitives. A closer look at exactly what causes the signature to be triggered should be taken before any decision is made. In either case, this is a very noisy signature, and should definitely be at least tuned.

## TFTP - External UDP connection to internal tftp server

TFTP connections from external hosts to internal TFTP servers probably should not happen. TFTP stands for Trivial File Transfer Protocol. It is a simplified FTP protocol, but should not be treated as an FTP. TFTP does not offer any security features, whereas FTP does. Allowing TFTP access to internal servers is a highly risky thing to do. Even only allowing access to specific files might be a problem since no authorization is performed.

It seemed rather unlikely that a TFTP server was running on MY.NET allowing external connections. Most TFTP servers exist on internal networks to allow diskless workstations to boot or within cable ISPs to allow their cable modems to retrieve the correct configuration. Most people who wish to allow others to retrieve files from them will run an FTP and give user accounts to individuals or allow anonymous access to the ftp.

To satisfy my curiosity in this TFTP traffic, I looked at the external addresses requesting the TFTP service. After looking at the sources and destinations, it seemed that most of the traffic was from addresses in MY.NET destined for 192.168.0.253. This address is a private IP address which is normally used on an internal network. It seems unlikely that anyone would spoof traffic to a TFTP server since it is used for file transfers, and a response would be necessary to receive any files. Instead, I believe that Snort picked it up because it was not configured to understand that 192.168.0.0/16 is an internal network, assuming that there is a subnetwork connected to MY.NET which uses that addressing scheme.

There are two actual external addresses which attempt to access TFTP on MY.NET. Those addresses are 63.251.39.161 and 63.210.198.194. Looking further into those addresses, I do a whois on 63.251.39.161 which returns as Kenneth Copeland Ministries. This seemed rather odd. I wanted further information about this host, so I attempted to access it through a browser hoping that it would be running a web server so that I could gain more information about the host who attempted to access a TFTP on MY.NET.

Accessing this server through a web browser had an interesting effect: I was prompted with a dialog telling me that I was accessing a file with a mime type of video/x-ms-asf. Now, I'm getting somewhere. This seems to be a streaming media server. I decided to attempt to access the other IP address in the same fashion to see if the results would agree. At this point, I was confronted with a web page labeled "PEC CardSaver Administration Panel." This was an interesting development. What is a PEC CardSaver? A quick search on Google answered that question for me. PEC stands for Parwan Electronics Corporation, and CardSaver is one of their products which is described as, "Voice over IP Pre-paid Calling." I found yet another form of streaming media. It would be my guess that both of these external hosts' streaming media servers had somehow created a connection which triggered the signature for a TFTP connection.

I'm glad that it looks as if there are no TFTP servers present accepting external connections on MY.NET, but further research should be done concerning the two external addresses. It should be confirmed that the streaming media is what causes that signature to trigger. Perhaps catching the traffic while it is in occurance on the network could be a possible next step.

*Top Ten Talkers -- Alerts*

Here is a list of the top ten source addresses based on all of the events found in the alert files over the period for our analysis.

| # | Src IP address | Total # | Unique Alerts | Dest. Addr. |
|---|---|---|---|---|
| 1 | MY.NET.84.151 | 31845 | 2 | 138 |
| 2 | 212.179.1.145 | 13901 | 1 | 1 |
| 3 | 217.136.73.54 | 6724 | 1 | 2 |
| 4 | 212.179.107.228 | 6055 | 1 | 4 |
| 5 | 80.200.225.161 | 5916 | 7 | 3 |
| 6 | MY.NET.111.235 | 4238 | 1 | 1 |
| 7 | MY.NET.111.232 | 4235 | 1 | 1 |
| 8 | MY.NET.111.219 | 4198 | 1 | 1 |
| 9 | MY.NET.111.231 | 4193 | 1 | 1 |
| 10 | MY.NET.111.230 | 4174 | 1 | 1 |

Now, here is the list of top ten destination addresses based on all of the events found in the alert files over the period of our analysis.

| # | Dest IP address | Total # | Unique Alerts | Src. Addr. |
|---|---|---|---|---|
| 1 | MY.NET.84.151 | 39955 | 9 | 141 |
| 2 | 192.168.0.253 | 21039 | 2 | 6 |
| 3 | MY.NET.113.4 | 15953 | 7 | 23 |
| 4 | 217.136.73.54 | 4841 | 1 | 1 |
| 5 | 80.200.225.161 | 4757 | 2 | 2 |
| 6 | MY.NET.88.193 | 4516 | 1 | 30 |
| 7 | MY.NET.105.204 | 3790 | 6 | 8 |
| 8 | MY.NET.84.193 | 3730 | 7 | 14 |
| 9 | 62.147.242.129 | 3141 | 1 | 1 |
| 10 | MY.NET.90.212 | 2858 | 1 | 4 |

Further analysis of these top talkers is necessary. Although it is expected that internal systems would talk to each other more than external systems would talk to us, that traffic should be valid traffic. Alerts generated through internal systems can be significant of trojans, worms, misconfigurations, and unauthorized services. External systems on this list should also be scrutinized. External systems are generally considered untrusted: a high number of events from them should be paid attention to. These high talkers are probably either a very noisy signature or a true attack of some kind. On another note, just because these systems are noisy, do not let them take your focus away from other important events. A web server can be defaced or otherwise compromised with only a few signatures catching the malicious events. On a production network, the signatures should be tuned to prevent noisy signatures from talking as much so that real attention can be paid to those events that represent true attacks.

Lets start with the external addresses listed in the source top talker list. The addresses we're looking at are 212.179.1.145, 217.136.73.54,

212.179.107.228, and 80.200.225.161.

212.179.1.145: A quick search in ACID can easily tell us that all of the traffic coming from this address or going to it is related to the Watchlist 000220 IL-ISDNNET-990517 signature. Looking back at that traffic, we remember that it's mostly peer to peer file sharing protocols, so lets move on.

217.136.73.54: Another search in ACID, and I have discovered that all traffic originating or destined from this address was labeled as High port 65535 tcp - possible Red Worm - traffic by Snort. This was also covered as one of the highest occuring detects. Since all of the traffic with this host alerted under the same signature, and that signature signifies the traffic produced by a pretty nasty worm, I would definitely check out the MY.NET hosts it connects to for infections with this particular worm. I will look further into this host under the external sources of interest section. If it is discovered that the hosts that it accesses on our network are infected with the Adore worm, an abuse email should be sent, the systems infected with the worm should be analyzed for further exploitation. Additionally, if one of those hosts were infected with the worm, an attacker was on the MY.NET network already. Further analysis of possible damage/exploitation on the network should be done.

212.179.107.228: Yet again, I do a search for this IP address in the list of alerts in ACID and discover that there is only one signature which alerts for this host. That signature is Watchlist 000220 IL-ISDNNET-990517. For this source, I discovered that all of the source ports were port 80. After discovering this, I checked to see if that host indeed runs a web server. Loading that address in my web browser produced a page with only the text, "w5.incredimail.com." Doing a nameserver lookup on that domain name gives me 212.179.107.241 which is most probably on the same network. I then check www.incredimail.com which comes out to 212.179.107.226. Seeing that incredimail owns IP addresses on either side of the one producing the alerts, it's safe to assume that incredimail also owns the one that did produce the alerts. It is probable that those alerts were nothing but web traffic being triggered by the watchlist.

80.200.225.161: This time ACID reports multiple signatures for this address. These signatures are: High port 65535 tcp - possible Red Worm - traffic, Null scan!, NMAP TCP ping!, SUNRPC highport access!, Probable NMAP fingerprint attempt, EXPLOIT x86 NOPS, and TFTP - External TCP connection to internal tftp server. These look like some alarming events, so we will look into this source some more. Most of these alerts are informational in nature, and most of them access ports 1 and 135. The TFTP connections are interesting because the ones seen on UDP ports did not look malicious. These are associated with an IP address that looks like it has been doing some scanning and may have accessed a Red worm infected host. Most of the traffic from this host falls under the Red worm signature. It would be wise to look into the destinations for any Red worm activity originating from this host to see if they are infected. This host will be further looked into under

external sources of interest.

This is the end of the list of top talkers for the sources, but there are still some destinations which are external to MY.NET which are external. Lets take a look at what systems have been noisy destinations. The destinations we see are: 192.168.0.253, 217.136.73.54, 80.200.225.161, and 62.147.242.129. 192.168.0.253 is a private address which we have already discovered is the source for a lot of TFTP traffic, so we strike this address from the list.

217.136.73.54 and 80.200.225.161: This address has already been talked about as a source address. Looking at ACID again, the same signatures are seen from when they were talked about as source addresses. The state must not have been considered or was lost when Snort recorded the events from these sources.

62.147.242.129: This host only appears with one signature: High port 65535 tcp - possible Red Worm - traffic. After looking into the alerts which have this host as a source or a destination, all of them have a common trait. They all are connecting to MY.NET.84.151. MY.NET.84.151 is also the number one source IP address listed in the top talkers lists, so this destination host should definitely be paid attention to. I will look further into this host under the external sources of interest section. If it is discovered that the hosts that it accesses on our network are infected with the Adore worm, an abuse email should be sent, the systems infected with the worm should be analyzed for further exploitation.

### *Top Ten Talkers -- Scans*

Here is a list of the top ten source addresses based on all of the events found within the scans files over the period of five days.

| # | Src IP address | Total # | Unique Ports |
|---|---|---|---|
| 1 | MY.NET.84.147 | 1234942 | 998 |
| 2 | MY.NET.83.146 | 930848 | 104 |
| 3 | MY.NET.70.176 | 602949 | 25 |
| 4 | MY.NET.150.213 | 342787 | 92 |
| 5 | MY.NET.91.72 | 264113 | 24 |
| 6 | MY.NET.114.45 | 238438 | 147 |
| 7 | MY.NET.91.252 | 163525 | 1652 |
| 8 | MY.NET.88.242 | 102531 | 3976 |
| 9 | MY.NET.118.6 | 84339 | 650 |
| 10 | MY.NET.106.228 | 60401 | 7 |

Below is the listing for the top ten destination addresses based on the scans found within all of the scans files retrieved from incedents.org for the five day period.

| # | Dst IP address | Total # | Unique Ports |
|---|---|---|---|
| 1 | MY.NET.150.210 | 14529 | 12822 |

| | | | |
|---|---|---|---|
| 2 | 66.177.41.202 | 3358 | 2 |
| 3 | 217.235.4.236 | 3309 | 1 |
| 4 | 164.77.214.242 | 2027 | 190 |
| 5 | 66.31.141.55 | 1948 | 2 |
| 6 | 61.103.141.201 | 1915 | 1 |
| 7 | 66.167.13.218 | 1899 | 1646 |
| 8 | 24.82.159.44 | 1830 | 1 |
| 9 | 24.85.157.182 | 1817 | 1 |
| 10 | 205.207.184.86 | 1675 | 3 |

While I am not going to further investigate these scans, it is interesting to note the number of unique ports from sources and those targeted on destination addresses. Also note that the scans were more spread out over destination addresses whereas with source addresses, a trend can be seen that the higher number of scans were generated all from addresses within MY.NET.

### *Top Ten Talkers -- OOS*

Here is a list of the top ten source addresses based on all of the events found within the Out of Spec files over the five day period.

| # | Src IP address | Total # | Unique Flags |
|---|---|---|---|
| 1 | 209.191.132.40 | 632 | 2 |
| 2 | 148.63.115.208 | 558 | 1 |
| 3 | MY.NET.70.183 | 462 | 1 |
| 4 | 133.11.36.54 | 298 | 1 |
| 5 | MY.NET.53.10 | 295 | 1 |
| 6 | 65.214.36.150 | 283 | 1 |
| 7 | 202.156.131.251 | 274 | 1 |
| 8 | 66.189.101.206 | 241 | 86 |
| 9 | 66.140.25.156 | 95 | 1 |
| 10 | 209.47.251.30 | 78 | 1 |

Now, here is the listing of the top ten destination addresses for Out of Spec packets found within the Out of Spec files for the five day period analyzed.

| # | Dst IP address | Total # | Unique flags |
|---|---|---|---|
| 1 | MY.NET.6.40 | 1323 | 3 |
| 2 | MY.NET.1.4 | 757 | 1 |
| 3 | MY.NET.117.143 | 632 | 2 |
| 4 | MY.NET.153.178 | 558 | 1 |
| 5 | MY.NET.130.12 | 298 | 1 |
| 6 | MY.NET.117.10 | 274 | 1 |
| 7 | MY.NET.84.193 | 247 | 86 |
| 8 | MY.NET.84.147 | 141 | 2 |
| 9 | MY.NET.99.85 | 130 | 1 |
| 10 | MY.NET.88.94 | 103 | 9 |

A lot could be said about all of these packets, and there is a lot that can be inferred from the low number of unique flag sets that tend to be used specific

to each source or destination. As I was gathering the information above, I could not help but to notice that many of the ports that the out of spec packets were destined for were Gnutella and Kazaa ports. It seems that not only do file sharing applications have no regard for bandwidth usage and copyright laws, but they also create a lot of anomolous traffic. Perhaps peer to peer applications should be treated with the same methods as trojan horses and worms: they produce all of the same characteristics and often do result in infections with trojan horses, worms, and viruses.

### *External Sources of Interest*

The whois for 217.136.73.54 follows. It looks as if this host is connecting to a host which might be compromised by the Adore worm. If the MY.NET host is proven to be infected with the Adore worm, the abuse contact listed within this whois should be contacted and sent an abuse email.

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      217.136.0.0 - 217.136.127.255
netname:      BE-SKYNET-ADSL1
descr:        Belgacom Skynet SA/NV
descr:        ADSL Access
country:      BE
admin-c:      SN2068-RIPE
tech-c:       SN2068-RIPE
rev-srv:      ns.ripe.net
rev-srv:      ns1.skynet.be
rev-srv:      ns2.skynet.be
rev-srv:      ns3.skynet.be
rev-srv:      ns4.skynet.be
status:       ASSIGNED PA
mnt-by:       SKYNETBE-MNT
changed:      ripe@skynet.be 20021125
source:       RIPE

role:         Skynet NOC administrators
address:      Belgacom Skynet  SA/NV
address:      rue colonel Bourg 124
address:      B-1140 Brussels
address:      Belgium
phone:        +3227061311
fax-no:       +3227269311
email:        ripe@skynet.be
admin-c:      JFS1-RIPE
tech-c:       PDH16-RIPE
nic-hdl:      SN2068-RIPE
remarks:      ----------------------------------------
remarks:      Abuse notifications to: abuse@skynet.be
remarks:      Network problems to: noc@skynet.be
remarks:      Peering requests to: peering@skynet.be
remarks:      ----------------------------------------
```

```
notify:       noc@skynet.be
mnt-by:       SKYNETBE-MNT
changed:      ripe@skynet.be 20010907
source:       RIPE
```

Below is the whois information for 62.147.242.129. It looks as if this host is also connecting to a MY.NET which might be compromised by the Adore worm. If the MY.NET host is proven to be infected with the Adore worm, the abuse contact listed within this whois should be contacted and sent an abuse email.

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      62.147.79.0 - 62.147.255.255
netname:      FR-PROXAD-DIALUP
descr:        Proxad / Free Telecom
descr:        Dynamic pool (dialup)
descr:        NCC#2002110278 (45312/45824)
country:      FR
admin-c:      ACP23-RIPE
tech-c:       TCP8-RIPE
status:       ASSIGNED PA
mnt-by:       PROXAD-MNT
changed:      tom@proxad.net 20021118
changed:      tom@proxad.net 20021125
source:       RIPE

route:        62.147.0.0/16
descr:        ProXad network / Free SA
descr:        Paris, France
origin:       AS12322
notify:       ripe-notify@proxad.net
mnt-by:       PROXAD-MNT
changed:      nhyvernat@corp.free.fr 20010913
source:       RIPE

role:         Administrative Contact for ProXad
address:      Free SA / ProXad
address:      24, rue Emile Menier
address:      75116 Paris
phone:        +33 1 56 26 20 00
fax-no:       +33 1 49 04 48 71
e-mail:       hostmaster@proxad.net
trouble:      Information: http://www.proxad.net/
trouble:      Spam: mailto:abuse@proxad.net
admin-c:      RA999-RIPE
tech-c:       NH1184-RIPE
tech-c:       TP684-RIPE
tech-c:       NS496-RIPE
nic-hdl:      ACP23-RIPE
notify:       ripe-notify@proxad.net
mnt-by:       PROXAD-MNT
changed:      nhyvernat@corp.free.fr 20010809
```

```
changed:      tom@proxad.net 20021125
source:       RIPE

role:         Technical Contact for ProXad
address:      Free SA / ProXad
address:      24, rue Emile Menier
address:      75116 Paris
phone:        +33 1 56 26 20 00
fax-no:       +33 1 49 04 48 71
e-mail:       hostmaster@proxad.net
trouble:      Information: http://www.proxad.net/
trouble:      Spam: mailto:abuse@proxad.net
admin-c:      RA999-RIPE
tech-c:       NH1184-RIPE
tech-c:       TP684-RIPE
tech-c:       NS496-RIPE
nic-hdl:      TCP8-RIPE
notify:       ripe-notify@proxad.net
mnt-by:       PROXAD-MNT
changed:      nhyvernat@corp.free.fr 20020626
changed:      tom@proxad.net 20021125
source:       RIPE
```

The following is the whois information for 80.200.225.161. This host also connected to a MY.NET host which may be infected with the Red worm which is also known as Adore. If the MY.NET host is proven to be infected with this worm, the ISP should be contacted with an abuse email. Furthermore, this host also alerted to other scan activity to ports 1 and 135 which should be detailed in the abuse letter.

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      80.200.0.0 - 80.200.255.255
netname:      BE-SKYNET-20011108
descr:        ADSL Customers
descr:        Skynet Belgium
country:      BE
admin-c:      JFS1-RIPE
tech-c:       PDH16-RIPE
status:       ASSIGNED PA
mnt-by:       SKYNETBE-MNT
changed:      ripe@skynet.be 20011212
source:       RIPE

route:        80.200.0.0/15
descr:        SKYNETBE-CUSTOMERS
origin:       AS5432
notify:       noc@skynet.be
mnt-by:       SKYNETBE-MNT
changed:      noc@skynet.be 20011116
source:       RIPE

person:       Jean-Francois Stenuit
```

```
address:      Belgacom Skynet NV/SA
address:      Rue Carli 2
address:      B-1140 Bruxelles
address:      Belgium
phone:        +32 2 706-1311
fax-no:       +32 2 706-1150
e-mail:       jfs@skynet.be
nic-hdl:      JFS1-RIPE
remarks:      ----------------------------------------
remarks:      Network problems to: noc@skynet.be
remarks:      Peering requests to: peering@skynet.be
remarks:      Abuse notifications to: abuse@skynet.be
remarks:      ----------------------------------------
mnt-by:       SKYNETBE-MNT
changed:      jfs@skynet.be 19970707
changed:      ripe@skynet.be 20021125
source:       RIPE

person:       Pieterjan d'Hertog
address:      Belgacom Skynet sa/nv
address:      2 Rue Carli
address:      B-1140 Brussels
address:      Belgium
phone:        +32 2 706 13 11
fax-no:       +32 2 706 13 12
e-mail:       piet@skynet.be
nic-hdl:      PDH16-RIPE
remarks:      ----------------------------------------
remarks:      Network problems to: noc@skynet.be
remarks:      Peering requests to: peering@skynet.be
remarks:      Abuse notifications to: abuse@skynet.be
remarks:      ----------------------------------------
mnt-by:       SKYNETBE-MNT
changed:      jfs@skynet.be 19990415
changed:      piet@skynet.be 19991210
changed:      piet@skynet.be 20000302
changed:      piet@skynet.be 20020329
source:       RIPE
```

This is the whois of 12.91.164.102. While This address was not the source or destination of any signature I have talked about so far, it did access a MY.NET IP address on the spooler port(515) several times. The spooler port is known to be a commonly scanned port and it may be vulnerable to exploits. **http://www.securiteam.com/exploits/5DQ0G000JA.html** describes an HP remote denial of service in which Hewlet Packard printers are vulnerable. If access to this spooler is not authorized from this host, an abuse letter should be sent to the contact listed in the whois information.

```
OrgName:    AT&T WorldNet Services
OrgID:      ATTW

NetRange:   12.0.0.0 - 12.255.255.255
CIDR:       12.0.0.0/8
NetName:    ATT
NetHandle:  NET-12-0-0-0-1
Parent:
NetType:    Direct Allocation
```

```
NameServer: DBRU.BR.NS.ELS-GMS.ATT.NET
NameServer: DMTU.MT.NS.ELS-GMS.ATT.NET
NameServer: CBRU.BR.NS.ELS-GMS.ATT.NET
NameServer: CMTU.MT.NS.ELS-GMS.ATT.NET
Comment:    For abuse issues contact abuse@att.net
RegDate:    1983-08-23
Updated:    2002-08-23

TechHandle: DK71-ARIN
TechName:   Kostick, Deirdre
TechPhone:  +1-919-319-8249
TechEmail:  help@ip.att.net

OrgAbuseHandle: ATTAB-ARIN
OrgAbuseName:   ATT Abuse
OrgAbusePhone:  +1-919-319-8130
OrgAbuseEmail:  abuse@att.net

OrgTechHandle: ICC-ARIN
OrgTechName:   IP Customer Care
OrgTechPhone:  +1-888-613-6330
OrgTechEmail:  qhoang@att.com

OrgTechHandle: IPSWI-ARIN
OrgTechName:   IP SWIP
OrgTechPhone:  +1-888-613-6330
OrgTechEmail:  swipid@nipaweb.vip.att.net

# ARIN Whois database, last updated 2003-01-21 20:00
# Enter ? for additional hints on searching ARIN's Whois database.
```

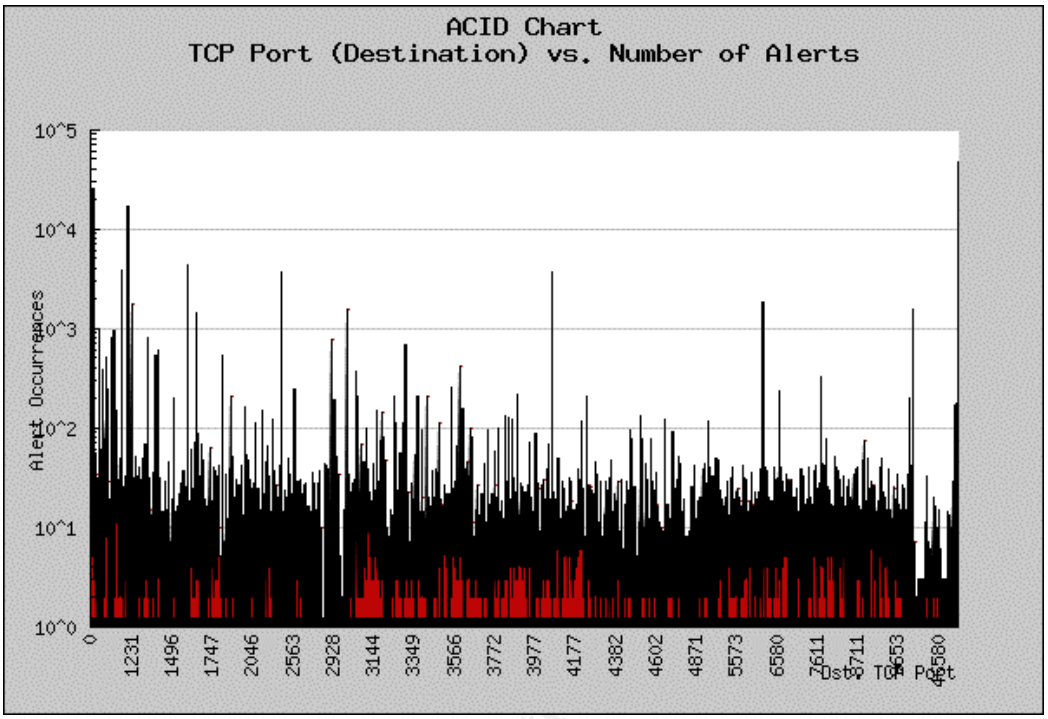
Here is another interesting host: 68.33.105.77. This host is the source for numberous signatures including: External FTP to HelpDesk MY.NET.70.49(15 times), External FTP to HelpDesk MY.NET.70.50(14 times), spp_http_decode: IIS Unicode attack detected(410 times), TFTP - External TCP connection to internal tftp server(160 times), spp_http_decode: CGI Null Byte attack detected(8 times), External POP to HelpDesk MY.NET.70.49(349 times), PHF attempt(12 times), External POP to HelpDesk MY.NET.70.50(276 times), and External RPC call(113 times). The address 68.33.105.77 resolves to pcp02102752pcs.towson01.md.comcast.net. The number of alerts for this host is unusual, so lets find out who we can contact if we find out that any of these alerts are part of a successful attack.

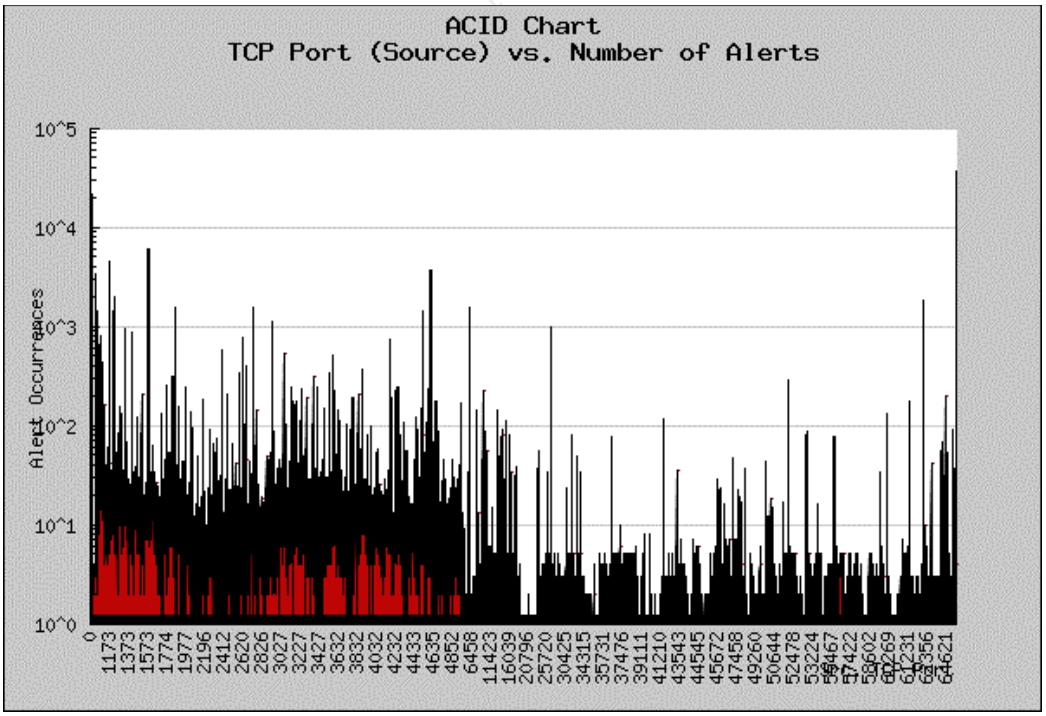
```
Comcast Cable Communications, Inc. JUMPSTART-1 (NET-68-32-0-0-1)
                                  68.32.0.0  - 68.63.255.255
Comcast Cable Communications, Inc. JUMPSTART-BALTIMOR-B1 (NET-68-33-
0-0-1)
                                  68.33.0.0  - 68.34.127.255

# ARIN Whois database, last updated 2003-01-21 20:00
# Enter ? for additional hints on searching ARIN's Whois database.
```

While this information does not directly give a way to send any abuse email, going to comcast's web site provides a way.

*Graphical Comparison of Source and Destination TCP Ports*



These two graphs illustrate the differences in the port ranges and numbers of alerts generated per port. Source hosts tend to have a much broader range of ports used than destinations. The number of alerts within the range from port 0 to around port 5000 are about the same between destinations and sources, however source addresses tend to have more alerts generated in the higher port ranges since source ports are not static.

What is interesting about these graphs is that the shape of the graphs in the areas that both the source and destination share the port range of 0-5000 seems to be about the same. Both have a higher number of alerts closer to port 0, but the number of alerts dips down towards the area around port 2500. The alerts per port then pick back up around port 3000 and then around port 4000, they dip back down again until they are very low around 5000. This similarity in the graphs may be caused by Snort only seeing the direction that each packet is going and not the direction the connection was made. Perhaps if Snort could track connections, a different trend would be seen.

### *Insights into Internal Machines*

Hosts with Adore worm traffic destined for them:

| | | |
|---|---|---|
| MY.NET.84.151 | MY.NET.88.193 | MY.NET.104.204 |
| MY.NET.108.34 | MY.NET.198.220 | MY.NET.91.252 |
| MY.NET.114.88 | MY.NET.88.165 | MY.NET.150.215 |
| MY.NET.6.40 | MY.NET.87.7 | MY.NET.118.6 |
| MY.NET.84.193 | MY.NET.91.104 | MY.NET.113.4 |
| MY.NET.150.16 | MY.NET.29.3 | |

While I am sure that not all of these hosts are infected with the Adore worm, the amount of traffic that some of them produced under the Red worm signature cannot be ignored. I would look into each of these systems, paying special regard to those which produced more Red worm alerts to determine if any of them truly are infected.

Another bit of information that caught my attention is that there were some hosts connecting to IRC which triggered XDCC bot signatures in Snort. While XDCC bots are not a 100% guarantee that the host running the bot has been owned, it can be a good indicator. Doing a listing of unique IP addresses associated with those alerts in ACID quickly came up with a list of internal address which were running the XDCC bots.

| | |
|---|---|
| MY.NET.88.168 | MY.NET.150.5 |
| MY.NET.105.48 | MY:NET.198.220 |

## *Defensive Recommendations*

As I stated in Detect #3, IRC and peer to peer file sharing applications may not be something you can block access to on your network, but if they must be run, it would be a good idea to separate those hosts which must access such services from the trusted network. An additional firewall and IDS should be put in place where the two networks join in an effort to protect the trusted network from the untrusted network.

There seems to be quite a bit of traffic triggering the Red worm signature. Research should be done on those internal hosts who are related to those alerts. The internal hosts which are associated with the traffic are lited above in the Insights into Internal Machines section.

Another thing I noticed, but did not find a place to discuss in this paper was some traffic entering MY.NET destined for hosts which are not a part of MY.NET. Some of this traffic was UDP and some was TCP. The IP address ranges which this traffic was originating from or destined for were all fairly close. It could be possible that the traffic was spoofed from a machine within MY.NET, but it would not be able to establish a TCP connection to any host, nor would it be able to receive any responses to UDP traffic. This could be significant of a misconfigured host on MY.NET.

## *References*


Fearnow, Matt. "SANS Institute: Adore Worm." April 12, 2001. URL: http://www.sans.org/y2k/adore.htm.

Dell, Anthony. "Adore Worm - Another Mutation." April 6, 2001. URL: http://www.sans.org/rr/threats/mutation.php.

Fearnow, Matt. "Adore Worm Detection and Removal Tool." Jan. 7 2003. URL:http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm.

Fyodor. "Re: [Snort-users] What is 'SMB Name Wildcard'." Aug 20 2000. URL: http://archives.neohapsis.com/archives/snort/2000-08/0289.html.

Rock, John. "HP printers vulnerable to remote DoS (spooler port)." April 26 2000. URL:http://www.securiteam.com/exploits/5DQ0G000JA.html.

Beardsley, Tod. "Intrusion Detection and Analysis: Theory, Techniques, and Tools." March 15, 2002. URL: http://www.giac.org/practical/Tod_Beardsley_GCIA.doc.

Drew, Steven. “Detection In Depth Practical Assignment.” 2001. URL: http://www.giac.org/practical/Steven Drew_GCIA.doc.

## Appendix A: Tools used for "Analyze This!"

While I used some scripts from other students' practicals during the analysis process for the scans files as well as the out of spec files, I also wrote my own PHP script for importing the Snort alert log format into a Snort style database. This made the process of analyzing all of the alerts much easier since ACID's web interface to the Snort database has many of the searching and sorting features I needed already built into it.

Below is a listing of all of the scripts I used as well as a short description of what the script was used for. I appologize for the lack of comments within the source code.

I should also note that to get the PHP Snort alert log parser working, I had to modify several variables in php.ini concerning the size of files to allow submitted in forms as well as memory and time allowed for the processing of a PHP script. These are not things that should be done on a production web server or any web server that receives traffic from untrusted sources. The PHP scripts should probably be converted into Perl so that form submissions would not be necessary and time spent processing would not be an issue.

This is just a simple html form that I used to submit my logs to the alert log parser with.

```
<form enctype="multipart/form-data" action="parselog.php" method="post">
  <input type="hidden" name="MAX_FILE_SIZE" value="100000000">
  Send this file: <input name="snortLog" type="file">
  <input type="submit" value="Send File">
</form>
```

Below is parselog.php which is used to accept the posted file and parse it out.

```
<?php
require("snortinserts.php");
//  file /home/httpd/htdocs/parselogs/snortlogs/alert.log
print "Parsing ".$_FILES['snortLog']['name']."...<br> \n";

if(is_uploaded_file($_FILES['snortLog']['tmp_name']))
{                                       // ****
  copy($_FILES['snortLog']['tmp_name'], "/home/httpd/htdocs/parselogs/snortlogs/alert.log");
}
else
{
  if(empty($_FILES['snortLog']['name']))
  {
    echo "Please select a file to upload first!";
  }
  else
  {
    echo "Possible file upload attack. Filename: ".$_FILE S['snortLog']['name'];
    echo "<br>\nYour IP has been logged!<br>\n";
  }
  exit;
}

clearstatcache();
if(file_exists("/home/httpd/htdocs/parselogs/snortlogs/alert.log")) // ****
{
  $fd=fopen ("/home/httpd/htdocs/parselogs/snortlogs/alert.log", "r"); //  ****
}
else
{
  exit;
}

/*****************************************************************
  Start Parsing Section
   - Modify this section to add support for other log formats.
*****************************************************************/

$logBuffer=fgets($fd, 4096);                    // get one line

/*****************************************************************
  Session Variables
   - Need stuff in here that Ref may need to clear up after
       parsing(like which team is which, and player ident)
   - Also need anything that will be added to the DB
*****************************************************************/

while(!feof($fd))
{
  if(ord($logBuffer[0])==0 || ord($logBuffer[0])==10 || ord($logBuffer[0])==13)
  {
    $logBuffer=fgets($fd, 4096);                // get one line
    continue;
  }
  if(preg_match("/spp_portscan/", $logBuffer, $match))
  {
    $logBuffer=fgets($fd, 4096);                // get one line
    continue;
  }
```

```
  {
//    printf("Matched: ".$match[1]."<br>\n");
    $proto=0;
  } else if(preg_match("/(?i)(udp)/", $logBuffer, $match))
  {
//    printf("Matched: ".$match[1]."<br>\n");
    $proto=1;
  } else {
//    printf("No match.  Assuming TCP<br>\n");
    $proto=2;
  }
  if(!preg_match("/(^[0-9][0-9])\/([0-9][0-9])-([0-9][0-9]):([0-9][0-9]):([0-9][0-9]).[0-9]*[ ]*\[\*\*\][
]([^][\*]+)[ ]\[\*\*\][ ]([0-9\.MYNET]*):([0-9]*)[ ]\-\>[ ]([0-9\.MYNET]*):([0-9]*)$/", $logBuffer,
$match))
  {
  if(!preg_match("/(^[0-9][0-9])\/([0-9][0-9])-([0-9][0-9]):([0-9][0-9]):([0-9][0-9]).[0-9]*[ ]*\[\*\*\][
]([^][\*]+)[ ]\[\*\*\][ ]([0-9\.MYNET]*)[ ]\-\>[ ]([0-9\.MYNET]*)$/", $logBuffer, $match))
  {
    printf("Malformed log file entry!: \n");
    printf("$logBuffer<br>\n");
    $logBuffer=fgets($fd, 4096);                    // get one line
    continue;
//    exit();
  }else{
    $proto=0;
  }
  }
  if(!empty($proto))
  {
    list( , $month, $day, $hours, $min, $sec, $signature, $src,  $sport, $dst, $dport)=$match;
  } else {
    list( , $month, $day, $hours, $min, $sec, $signature, $src, $ds t)=$match;
  }
  $date = "03".$month.$day.$hours.$min.$sec;
  switch($proto)
  {
    case 0: //icmp
      inserticmp($date, $signature, $src, $dst);
      break;
    case 1: //udp
      insertudp($date, $signature, $src, $sport, $dst, $dport);
      break;
    case 2: //tcp
      inserttcp($date, $signature, $src, $sport, $dst, $dport);
      break;
  } // end switch
  $logBuffer=fgets($fd, 4096);                    // ge t one line
  //print $logBuffer."<br>\n";
} // end while
print "done.<br>\n";
/*************************************************************
  End Parsing Section
*************************************************************/
fclose($fd);
unlink("/home/httpd/htdocs/parselogs/snortlogs/alert.log");
?>
```

Next is snortinserts.php.  It actually inserts the data to the database.

```
<?php
if(!defined("_SNORT_INCLUDE_")){
define("_SNORT_INCLUDE_",1);

include("mysql.php");

//return cid
function getlastcid($sid)
{
  $db = new db();
  $db->open("snort", "localhost", "snort", "snort");
  $sql = "select  last_cid from sensor where sid=$sid";
  //print "$sql ".date("H:i:s")." <br> \n";
  $query=new query($db, $sql);
  $query->getrow();
  $cid=$query->field(0);
  $cid++;
  $sql = "update sensor set last_cid=$cid where sid=$sid";
  $query=new query($db, $sql,1 );
  return $cid;
}

function insertevent($sid,$cid,$sig,$date)
{
  $db = new db();
  $db->open("snort", "localhost", "snort", "snort");
  $sql = "insert into event (sid, cid, signature, timestamp) values ($sid, $cid, $sig, '$date')";
  //print "$sql ".date("H:i:s")." <br>\n";
  $query=new query($db, $sql, 1);
}

function insertiphdr($sid,$cid,$src,$dst,$proto)
{
  if(!preg_match("/([MYNET0-9]*)\.([MYNET0-9]*)\.([MYNET0-9]*)\.([MYNET0-9]*)/", $src,
$match))
    print "Bad src IP!! \"$src\"<br>\n";
  list( , $one, $two, $three, $four)=$match;
  if($one == "MY") $one = 0;
  if($two == "NET") $two = 0;
  $src_int32 = sprintf("%2x",$one);
  $src_int32 .= sprintf("%2x",$two);
  $src_int32 .= sprintf("%2x",$three);
  $src_int32 .= sprintf("%2x",$four);
  $src_int32 = preg_replace("/\s/", "0",$src_int32);
  $ip_src = hexdec($src_int32);

  if(!preg_match("/([MYNET0-9]*)\.([MYNET0-9]*)\.([MYNET0-9]*)\.([MYNET0-9]*)/", $dst,
$match))
  print "Bad dst IP!! \"$dst\"<br>\n";
  list( , $one, $two, $three, $four)=$match;
  if($one == "MY") $one = 0;
  if($two == "NET") $two = 0;
  $dst_int32 = sprintf("%2x",$one);
  $dst_int32 .= sprintf("%2x",$two);
  $dst_int32 .= sprintf("%2x",$three);
  $dst_int32 .= sprintf("%2x",$four);
  $dst_int32 = preg_replace("/\s/", "0",$dst_int32);
  $ip_dst = hexdec($dst_int32);
```

```
  //print "$ip_src<br>\n";
  //print "$ip_dst<br>\n";
  $db = new db();
  $db->open("snort", "localhost", "snort", "snort");
  $sql = "insert into iphdr (sid, cid, ip_src, ip_dst, ip_proto) values ($sid, $cid, $ip  _src, $ip_dst,
$proto)";
  //print "$sql ".date("H:i:s")." <br> \n";
  $query=new query($db, $sql, 1);
}

function insertudphdr($sid,$cid,$sport,$dport)
{
  $db = new db();
  $db->open("snort", "localhost", "snort", "snort");
  $sql = "insert into udphdr (s id, cid, udp_sport, udp_dport) values ($sid, $cid, $sport, $dport)";
  //print "$sql ".date("H:i:s")." <br> \n";
  $query=new query($db, $sql, 1);
}

function inserttcphdr($sid,$cid,$sport,$dport)
{
  $db = new db();
  $db->open("snort", "localhost", "snort ", "snort");
  $sql = "insert into tcphdr (sid, cid, tcp_sport, tcp_dport) values ($sid, $cid, $sport, $dport)";
  //print "$sql ".date("H:i:s")." <br> \n";
  $query=new query($db, $sql, 1);
}

function inserticmphdr($sid,$cid)
{
  $db = new db();
  $db->open("snort", "localhost", "snort", "snort");
  $sql = "insert into icmphdr (sid, cid) values ($sid, $cid)";
  //print "$sql ".date("H:i:s")." <br> \n";
  $query=new query($db, $sql, 1);
}
```

```
function inserticmp($date, $signature, $src, $dst)
{
  $cid=getlastcid(1);
  $sig=dosignature($signature);
  insertevent(1,$cid,$sig,$date);
  insertiphdr(1,$cid,$src,$dst,1);
  inserticmphdr(1,$cid);
}

function insertudp($date, $signature, $src, $sport, $dst, $dport)
{
  $cid=getlastcid(1);
  $sig=dosignature($signatu re);
  insertevent(1,$cid,$sig,$date);
  insertiphdr(1,$cid,$src,$dst,17);
  insertudphdr(1,$cid,$sport,$dport);
}

function inserttcp($date, $signature, $src, $sport, $dst, $dport)
{
  $cid=getlastcid(1);
  $sig=dosignature($signature);
  insertevent(1,$c id,$sig,$date);
  insertiphdr(1,$cid,$src,$dst,6);
  inserttcphdr(1,$cid,$sport,$dport);
}

}
?>
```

The scripts use the Muse database abstraction, but I've modified it slightly for my purposes. The modified version is listed below.

```
<?php
if(!defined("_DB_CLASS_")){
define("_DB_CLASS_",1);
/*
My version.
added error checking... errors don't get printed on screen. they are reported
back to the calling script.  Usefull when execu ting inserts and things that
don't require any data back except to know if it s ucceeded.
*/
/*******************************************************************************

  Abstract DB , MySQL module, version 2.0b3

  Copyright (C) 1998  Muze

  This program is free software; you can redistribute it and/or
  modify it under the terms of the GNU General Public License
  as published by the Free Software Foundation; either version 2
  of the License, or (at your option) any later version.

  This program is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRAN TY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program; if not,  write to the Free Software
  Foundation, Inc., 59 Temple Place  - Suite 330, Boston, MA  02111 -1307, USA.

*******************************************************************************
```

```
  for information,comments or bugreports, mail abstractdb@muze.nl

  Changelog:

  v2.0b3 22 jan. 1998
    - db->db() constructor now sets a type variable (db ->type) with the
      default value of 'database_type'.
    - new function query->error() which returns a description of the last
      mysql error.
    - changed db->nextid() to use autoincrement capabilities of mysql, code
      contributed by Brian Moon.
    - added check $this->result!=0 in query->getrow
    - added @ on mysql_data_seek in query ->firstrow

  v2.0b2 1 dec. 1998
    - fixed 2 small bugs in db->nextid() when db_sequence doesn't exist yet.

  v2.0b1 first version with the new interface.

*******************************************************************************/

class db {

  var $connect_id;
  var $type;

  function db($database_type="mysql") {
    $this->type="mysql";
    // dl("mysql");
  }

  function open($database="{database}", $host="{host}", $user="{user}", $password="{password}")  {
    $this->connect_id=mysql_pconnect($host, $user, $password);
    if ($this->connect_id) {
      $result=mysql_select_db($database);
      if (!$result) {
        mysql_close($this->connect_id);
        $this->connect_id=$result;
      }
    }
    return $this->connect_id;
  }

  function lock($table, $mode="write") {
  // FIXME: will this work for other databases,  must check
  // for now: mode maybe 'read' or 'write'

    $query=new query($this, "lock tables $table $mode");
    $result=$query->result;
    return $result;
  }

  function unlock() {
  // unlocks any and all tables which this process locked

    $query=new query($this, "unlock tables");
    $result=$query->result;
```

```
    return $result;
  }

  function nextid($sequence) {
  // Function returns the next available id for $sequence, if it's not
  // already defined, the first id will start at 1.
  // This function will create a table for each sequence called
  // '{sequence_name}_seq' in the current database.
  // Based on code by Brian Moon.

    $esequence=ereg_replace("","",$sequence)."_seq";
    $query=new query($this, "REPLACE INTO $esequence values (  ", nextval+1)");
    if ($query->result) {
      $nextid=mysql_insert_id($this->connect_id);
    } else {
      $query->query($this, "CREATE TABLE $esequence ( seq char(1)
DEFAULT '' NOT NULL,  nextval bigint(20) unsigned DEFAULT '0' NOT NU LL auto_increment,
PRIMARY KEY (seq), KEY nextval (nextval) )");
      // there's no way to check if a create table has succeeded except by trying to insert
      // a new value. Since you don't want an endless loop, a recursive call to
      // nextid should not be made:
      $query->query($this, "REPLACE INTO $esequence VAL UES ( '', nextval+1 )");
      if ($query->result) {
        $nextid=mysql_insert_id($this->connect_id);
      } else {
        $nextid=0;
      }
    }
    return $nextid;
  }

  function error() {
    return mysql_errno($this->connect_id).": ".mysql_error($this->connect_id);
  }

  function close() {

  // Closes the database connection and frees any query results left .

    if ($this->query_id && is_array($this->query_id)) {
      while (list($key,$val)=each($this->query_id)) {
        @mysql_free_result($val);
      }
    }
    $result=@mysql_close($this->connect_id);
    return $result;
  }

  function addquery($query_id) {
  // Function used by the constructor of query. Notifies the
  // this object of the existance of a query_result for later cleanup
  // internal function, don't use it yourself.

    $this->query_id[]=$query_id;
  }

};

/************************************ QUERY  ***************************/
```

```
class query {
  var $error;
  var $result;
  var $row;
  var $numfields;

  function query(&$db, $query="", $execonly=0) {
  // Constructor of the query object.
  // executes the query, notifies the db object of the query result to clean
  // up later
    if ($query) {
      if ($this->result) {
        $this->free(); // query not called as constructor therefore there may
                       // be something to clean up.
      }
      $this->result=mysql_query($query, $db->connect_id);
      $this->error=mysql_errno();
      $this->numfields = 0;
      if(empty($this->error) && !$execonly)
      {
        $db->addquery($this->result);
        $this->numfields = mysql_num_fields($this->result);
      }
    }
  }

  function getincrement() {
  // Gets the insert id for the last insert. (auto  increment value)
    return mysql_insert_id();
  }

  function geterror() {
  // Gets the next row for processing with $this ->field function later.
    return $this->error;
  }

  function getrow() {
  // Gets the next row for processing  with $this->field function later.

    if (empty($this->error) && !empty($this->numfields)) {
      $this->row=mysql_fetch_array($this->result);
      $this->error=mysql_errno();
    } else {
      $this->row=0;
    }
    return $this->row;
  }

  function num_fields() {
    return $this->numfields;
  }

  function field_name($field) {
//may need to add $this ->error stuff in later
    return mysql_field_name($this->result, $field);
  }
```

```
 function field($field) {
 // get the value of the field with name $field
 // in the current row

  $result=$this->row[$field];
  return $result;
 }

 function firstrow() {
 // return the current row pointer to the first row
 // (CAUTION: other versions may execute the query again!! (e.g. for oracle))

  $result=@mysql_data_seek($this->result,0);
  if ($result) {
   $result=$this->getrow();
  }
  return $this->row;
 }

 function nthrow($rownum) {
 // set the current row pointer to the nth row
  $result=@mysql_data_seek($this->result,$rownum);
  if($result) {
   $result=$this->getrow();
  }
  return $this->row;
 }
 function free() {
 // free the mysql result tables

  return @mysql_free_result($this->result);
 }

};

}
?>
```

Here are some of the scripts I used for processing the scans and oos files.

```
#!/usr/bin/perl -w
while(<>) {
      if (m/([0-9\.]+|MY\.NET\.[0-9]+\.[0-9]+):([0-9]+) -> ([0-9\.]+|MY\.NET\.[0-9]+\.[0-9]+):([0-
9]+)/)
      {
            $one_line = $_;
            chomp($one_line);
            $_ = <>;
            $_ = <>;
            @blah = split(/ /);
            print "$one_line  $blah[0]\n";
       }
}
```

This script grabs source and destination IP addresses and ports and the

packet flags which caused the packet to be determined as OOS, and formats it to one line per packet space delimted. A similar script is used for scans. It is borrowed from Mike Wisener's GCIA practical(recently submitted).

```
#!/bin/bash

# Find top sources ips
awk '{ print $1 }' oos.delimited | sort  -n | uniq -c | sort -r -n > oos.sourcecount

# Find top destination ips
awk '{ print $3 }' oos.delimited | sort  -n | uniq -c | sort -r -n > oos.destcount
```

This script was used to find all unique souce and destination IP addresses within the scans files. This script and the script below were taken from Steven Drew's GCIA practical, and has been modified slightly.

```
#!/bin/bash

# Find top sources ips
awk '{ print $1 }' oos.delimited | sort  -n | uniq -c | sort -r -n > oos.sourcecount

# Find top destination ips
awk '{ print $3 }' oos.delimited | sort  -n | uniq -c | sort -r -n > oos.destcount
```

The above script I used to find all unique source and destination IP addresses for the OOS files.

```
#!/bin/bash

#./scriptname <ip> <file> <source|dest> <count>

if [ "${3}" = "source" ]
then
  if [ "${4}" = "uniq" ]
  then
    grep -e "^$1\ [0-9]*" $2 | awk '{ print $2 }' | sort  -n | uniq -c | sort -rn | wc -l
  else
    grep -e "^$1\ [0-9]*" $2 | awk '{ print $2 }' | sort  -n | uniq -c | sort -rn | head -$4
  fi
else
  if [ "${4}" = "uniq" ]
  then
    grep -e " $1\ [0-9]*" $2 | awk '{ print $4 }' | sort  -n | uniq -c | sort -rn | wc -l
  else
    grep -e " $1\ [0-9]*" $2 | awk '{ print $4 }' | sort  -n | uniq -c | sort -rn | head -$4
  fi
fi
```

This script is what I used to retrieve some of the statistics about the oos and scans. It is not a polished script; I was modifying it constantly while using it.