GCIA (GIAC Certified Intrusion Analyst)
Practical Assignment v4.0

Chris Sia

December 24, 2004

Table of Contents

Abstract

The goal of this practical assignment is to demonstrate my ability to use the tools and tactics that have been created by those who have gone before me, to the level of proficiency that is required of an intrusion analyst. This paper outlines the general security stature of an organization and breaks down the various levels of responsibility that are vital to the security health of a network. The first portion of this practical is an executive summary. Geared to the executive mindset, it strives to convey the importance of these interlocking security components in a straightforward and clear format, devoid of confusing technical jargon.

The second portion of this practical is an in-depth decode of three traces picked from the binary snort alerts posted on http://isc.sans.org/logs/raw. In these decodes, I attempt to show how using only alerts, and not having an in-depth knowledge of the network, an analyst can use traces not only to determine what is being targeted, but also to glean information about the network and specific services running.

In this second section I also show various other metrics obtainable from the alert traces, as well as defensive recommendations based on not only the alerts I have chosen to decode, but on other observed traffic. It is part of any and all analyst's duties not only to be able to detect when malicious activity has taken place, but to respond to and mitigate the incident in an effective and efficient manner. This demonstrates the analyst's deep understanding of exactly what technically has occurred.

In the final section I walk the reader through the process I took to analyze the detects that I have decoded in this paper. I will discuss the initial steps taken to identify the alerts, and the thought process involved in selecting the three alerts included in this paper.

The next sequence will go into detail about the process taken to analyze the individual detects in depth to extract as much information from not only the alert packets themselves but other related and non-related network activity.

Part I – Executive Summary

The role of the security analyst in an organization is a very specialized one. Understanding how normal and malicious activity appears from the network perspective is not a skill many possess, not even many system Administrators. Dedicating the resources and time to such measures is usually overlooked and under-appreciated, yet is always the first thought on everyone's mind when something does go wrong.

In reviewing the logs generated from your perimeter Intrusion Detection Systems (IDS's), several key areas of interest should be noted. The logs revealed several known exploits and policy violations present in your internal network such as worm activity, Peer-To-Peer file sharing, and possible pornographic web browsing. Although the placement of IDS's is a key component of a sound network security architecture, many more improvements are necessary to ensure the future integrity of the network. Proper network security is a multi-tiered architecture in which each component is reliant on the other to function properly and efficiently. First of all, IDS logs are not the be all and end all of network security analysis. Although IDS logs play a vital role in determining if any "known" malicious activity is present on your network, they are not the security "silver bullet" that will keep your network safe from intruders or malicious logic. Although much can be gathered from the snort alerts that were reviewed in this paper, further investigation would be required to validate most of the conclusions, simply because the information is not present. Access to firewall logs, router/switch logs, web logs and syslog/eventlogs allow a more in-depth and thorough analysis. These logs will aid in the correlation of multiple events to piece together the entire story of an attack. Think of it this way, if you had a single shot camera situated outside the door to your home that was configured to snap a shot every time a suspicious person came to your porch, you would know who was possibly trying to break into your home. But what else would you know? Would you know if they successfully broke into your home, or if they even intended on breaking into your home? This would be your IDS, a system that simply flags packets based on pre-defined rules.

Let's say now you need to know everyone that made it into your home. You would need a second camera to capture this, right? Well, depending on where your IDS is placed this would be your firewall, or an additional IDS on the internal side of your firewall. Now, wouldn't you also like to see what they did once they got in? Of course, who wouldn't? This would require all sorts of other cameras placed at key locations snapping shots whenever anyone tried to access what the cameras were watching. These represent your syslog or NT event logs.

With all of these snapshots put together you create a complete story of who tried to get in, who got in, and what mischievous things they did after they got in. Now, let me add this, how would you feel about having a video camera follow and record everything that occurred both in and right outside your house, twentyfour hours a day? Now everything that happens is caught on film, and little is left to the imagination. These are very good things to have in the event that any legal matters should arise. Sounds great right, actually sounds like physical security measures already put in place for your company. Well, such measures for information security are called “full content data” (Bejtlich 119).

To ensure that as much information can be ascertained from the network without the analyst having to touch the systems on it, it is recommended that “full content data” be captured and analyzed on regular cycles. Allowing the analyst to see the “big picture” will not only speed up and make more thorough the analysis process, but it will also allow analysis on everything your IDS does not capture, such as possible malicious activity that has gone undetected for one reason or another. If full content data is not a feasible possibility due to the complexities of your network, then perhaps utilizing session data with tools such as Argus or Cisco Net-Flow would be a good alternative. This will at least allow security analysts to audit who is talking on your network, where and how they were talking, and how much was said. All of this data can provide many useful clues to troubleshoot and investigate security issues.

Lastly, having a staff of well-trained and dedicated security analysts is a must. The ones who are viewing the logs on a regular basis should know, or should be taught the many methods and procedures that go into security monitoring, as well as how known exploits appear from the network and packet perspective.

Network and computer security can be a hard concept for management to define. It is also intangible and impossible to predict. It is not only when the latest Microsoft worm hits everyone, or when a hacker is discovered perusing your businesses’ sensitive files that the security staff is hard at work, it’s also when end users and management have access to the resources they use an a daily basis to make critical business decisions amid the daily barrage of malicious activity circling the Internet. It is during these seemingly quiet times to end users that the network security analysts are at their peak, because only they know that the integrity of their network can change for the worse in a matter of seconds.

The types of measures I have described here, take time and resources to implement well, but setting a strong security foundation will save time and money in the long run.

Part II – Detailed Analysis

Detect #1 - Code Red worm.

Description of the Detect
This detect was analyzed from the http://isc.sans.org/logs/raw/2002.9.3, 2002.9.9, and 2002.9.10 snort alert logs. I reviewed several days' worth of logs to get a better understanding of what alerts were unique and what alerts were daily and ongoing. The method involved taking the pcap alert files and using a tool bundled with Ethereal called mergecap to merge all the individual pcap files into one. The command syntax is:

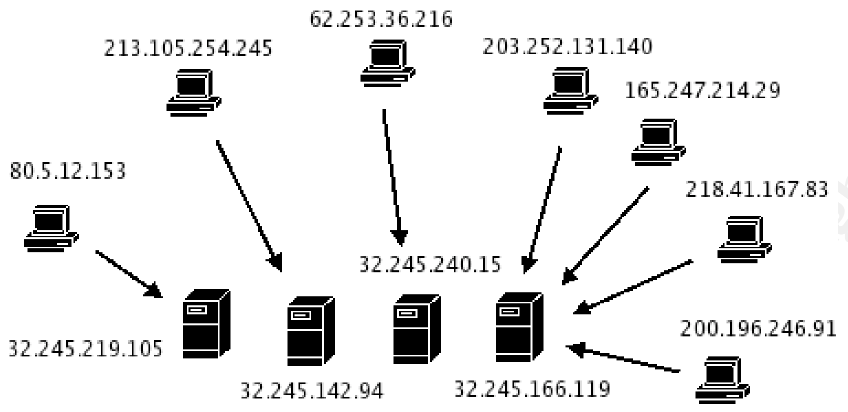
```
$ mergecap  -w {output file name} {pcap file1} {pcapfile2} ......
```

Once all three days had been merged, I ran the merged pcap file through snort and ethereal. While looking through the http_inspect alerts, this one jumped out:

```
[**] (http_inspect) BARE BYTE UNICODE ENCODING [**]
10/10-09:32:32.736507 200.196.246.91:4341 -> 32.245.166.119:80
TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:1504
***AP*** Seq: 0xD3384408  Ack: 0x531A4C30  Win: 0x7D78  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 10  .....3....&...E.
0x0010: 05 E0 00 00 00 00 F0 06 00 00 C8 C4 F6 5B 20 F5  ............[ .
0x0020: A6 77 10 F5 00 50 D3 38 44 08 53 1A 4C 30 50 18  .w...P.8D.S.L0P.
0x0030: 7D 78 00 00 00 00 47 45 54 20 2F 64 65 66 61 75  }x....GET /defau
0x0040: 6C 74 2E 69 64 61 3F 4E 4E 4E 4E 4E 4E 4E 4E 4E  lt.ida?NNNNNNNNN
0x0050: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
.........
0x0130: 00 00 00 00 00 00 C3 03 00 00 00 78 00 FA 20 25  ...........x.. %
0x0140: 75 39 30 39 30 25 75 36 38 35 38 25 75 63 62 64  u9090%u6858%ucbd
0x0150: 33 25 75 37 38 30 31 25 75 39 30 39 30 25 75 36  3%u7801%u9090%u6
0x0160: 38 35 38 25 75 63 62 64 33 25 75 37 38 30 31 25  858%ucbd3%u7801%
0x0170: 75 39 30 39 30 25 75 39 30 39 30 25 75 38 31 39  u9090%u9090%u819
0x0180: 30 25 75 30 30 63 33 25 75 30 30 30 33 25 75 38  0%u00c3%u0003%u8
0x0190: 62 30 30 25 75 35 33 31 62 25 75 35 33 66 66 25  b00%u531b%u53ff%
0x01A0: 75 30 30 37 38 25 75 30 30 30 30 25 75 30 30 3D  u0078%u0000%u00=
0x01B0: 61 20 20 48 54 54 50 2F 31 2E 30 0D 0A 43 6F 6E  a  HTTP/1.0..Con
0x01C0: 74 65 6E 74 2D 74 79 70 65 3A 20 74 65 78 74 2F  tent-type: text/
0x01D0: 78 6D 6C 0A 48 4F 53 54 3A 77 77 77 2E 77 6F 72  xml.HOST:www.wor
0x01E0: 6D 2E 63 6F 6D 0A 20 41 63 63 65 70 74 3A 20 2A  m.com. Accept: *
0x01F0: 2F 2A 0A 43 6F 6E 74 65 6E 74 2D 6C 65 6E 67 74  /*.Content-lengt
```

Here we see the default.ida? String followed by a series of N's which is indicative of the Code Red worm signature CVE-2001-0500. The alert logs flagged several packets with this signature, all originating from different external hosts. This is indicative of infected web servers attempting to spread the worm to uninfected hosts.

Once a web server is infected with Code Red, it will attempt to spread to other web servers via a buffer overrun. Each individual instance of Code Red wound up scanning the same IP addresses due to a bug in the code.



Code Red Link Graph

<u>Reason Detect was Selected</u>

Although in the days I chose to analyze the Code Red signatures were few, upon my initial review of all alerts generated I noticed a large majority of the traffic was web-related. Not knowing anything else about the network, I chose to look at one of the hardest hitting web based worms in history.

<u>Detect was generated by:</u>
The alert was generated by snort 2.2.0 running on Red Hat Linux kernel version 2.4.20-6. The snort http_inspect preprocessor caught the attack. The trace was also detected while running the raw packets through Ethereal. The interesting thing about the alert is that it was detected by the Snort http_inspect preprocessor and not a specific rule. The http_inspect preprocessor normalizes and inspects all defined http port traffic in an effort to detect http anomalies. The alert generated was non-specific and the actual Code Red worm packets were mixed in with other miscellaneous web traffic alerts. It was easier to detect these packets while viewing them in ethereal due to the default.ida?NNNNNNNNNNNNNNNN payload that stuck out like a sore thumb. To cross check my results, I downloaded and ran the binary alert file against snort 1.9.0. This had an interesting effect. The "WEB-IIS ISAPI .ida attempt" alert was triggered which matched the ".ida?" string with a data size of greater than 239 bytes.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS ISAPI .ida attempt"; uricontent:".ida?"; nocase; dsize:>239; flow:to_server,established; reference:arachnids,552; classtype:web-application-attack; reference:bugtraq,1065; reference:cve,CAN-2000-0071; sid:1243;  rev:6;)
```

<u>Probability the source address was spoofed:</u>
Unlikely. The Code Red worm relies on an established TCP connection to infect the victim host. Most likely these are real infected web servers scanning the internet for propagation opportunities.

Attack Mechanism:

The Code Red worm exploits a buffer overflow in the IIS indexing services IDQ.DLL. The attacking host sends a crafted GET request to the IIS default.ida file overrunning its buffer with the "NNNNNN…" payload.

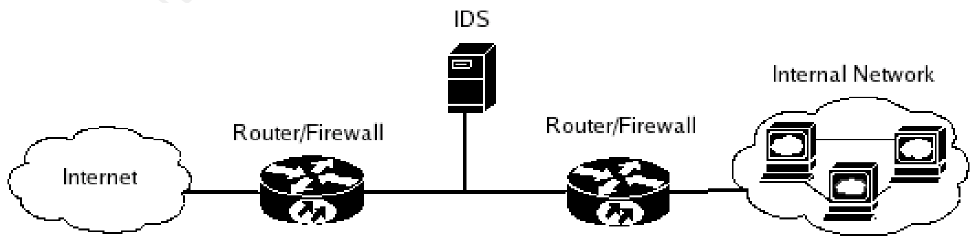
Correlations:

Of the four hosts that were targeted, only one had generated any logs that were traceable via snort alerts.

Below is a tcpdump output of a 403 Forbidden packet coming from the victim web server. It is clear in this example that the victim is an Apache web server running on Linux. (Note: the packet TTL of 63 and the packet data content.) This gives overwhelming evidence of the victim host running a non-vulnerable OS and web server for this particular exploit.

```
19:12:07.466507 32.245.166.119.80 > 212.62.35.225.1168: P [bad tcp cksum 1815!]
582037703:582038239(536) ack 1357129 win
32696 (DF) (ttl 63, id 672, len 576, bad cksum 6274!)
0x0000   4500 0240 02a0 4000 3f06 6274 20f5 a677      E..@..@.?.bt...w
0x0010   d43e 23e1 0050 0490 22b1 30c7 0014 b549       .>#..P..".0....I
0x0020   5018 7fb8 42e6 0000 4854 5450 2f31 2e31      P...B...HTTP/1.1
0x0030   2034 3033 2046 6f72 6269 6464 656e 0d0a       .403.Forbidden..
0x0040   4461 7465 3a20 5468 752c 2031 3020 4f63       Date:.Thu,.10.Oc
0x0050   7420 3230 3032 2030 343a 3032 3a31 3020        t.2002.04:02:10.
0x0060   474d 540d 0a53 6572 7665 723a 2041 7061        GMT..Server:.Apa
0x0070   6368 652f 312e 332e 3132 2028 556e 6978       che/1.3.12.(Unix
0x0080   2920 2028 5265 6420 4861 742f 4c69 6e75       )..(Red.Hat/Linu
0x0090   7829 2046 726f 6e74 5061 6765 2f34 2e30      x).FrontPage/4.0
......
```

Since this exploit is only known to affect Windows hosts running IIS it is clear that the victim (32.245.166.119) web server is not adversely affected. From this we can also identify the placement of the Snort IDS sensor. It is one hop away from the targeted web server.



To filter on all packets that had the Code Red exploit signature, I ran ngrep on the binary alert file looking for the string "default.ida". I have only included a portion of each packet that is relevant to this section.

```
$ ngrep -q -I <cap file> 'default.ida'

-203.252.131.140:1569 -> 32.245.166.119:80 [AP]  GET /default.ida?NNNNNNNNNNNNNN...
-80.5.120.153:1441 -> 32.245.219.105:80 [AP] 26524@0:1448 ...P+....."oP.Dp%...GET
/default.ida?NNNNNNNNNN...
-213.105.254.245:3617 -> 32.245.142.94:80 [AP] 2480@0:1448 .!.P.V.../.WP.Dp....GET
/default.ida?NNNNNNNNNN...
-165.247.214.29:1423 -> 32.245.166.119:80 [AP]  GET /default.ida?NNNNNNNNNNNNNN...
-218.41.167.83:61855 -> 32.245.166.119:80 [AP]  GET/default.ida?NNNNNNNNNNNNNN...
-200.196.246.91:4341 -> 32.245.166.119:80 [AP]  GET /default.ida?NNNNNNNNNNNNNN...
-62.253.36.216:1294 -> 32.245.240.15:80 [AP] 3031@0:1448 ...P..C=v...P."8....GET
/default.ida?NNNNNNNNNN...
```

As you can see there are four unique targeted hosts with seven attacking hosts. Only one of the target hosts can be determined to actually be running a web server due to the presence of return traffic in the form of 403 error messages (not Code Red related).

It is not possible for us to make any assumptions about the other three hosts targeted due to the lack of captured return traffic. It is possible due to this fact that they may be Windows hosts running IIS.

<u>Evidence of active targeting:</u>

This exploit scans attached networks for hosts with port 80 open. This scanning by infected hosts is random. The infected hosts do target port 80 specifically.

<u>Severity</u>

To determine the severity we use the equation:

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality = 4
The Code Red worm was one of the fastest spreading Internet worms of all time.

Lethality = 1
The Code Red worm did nothing to damage data, and it did not install any type of backdoor. The worst thing it did was at a given time perform a Denial of Service (DoS) attack against a specific website and scanning would slow down the infected systems. It ran in memory so most times a reboot would clean the system.

System Countermeasures = 5

The production web server is a Linux OS running Apache which is not vulnerable to the ISAPI buffer overflow.

Network Countermeasures = 3
It seems that the IDS system did have signatures for the Code Red worm so it was detectable. It is difficult to tell if the router past the IDS was filtering for this particular worm payload.

Severity = (4 + 1) - (5 + 3) = -3

Detect # 2 – Backdoor Q Access

<u>Description of the Detect</u>
This detect was analyzed from the http://isc.sans.org/logs/raw/2002.9.3, 2002.9.9, and 2002.9.10 snort alert logs. I reviewed several days' worth of logs to get a better understanding of what alerts were unique and what alerts were daily and ongoing.

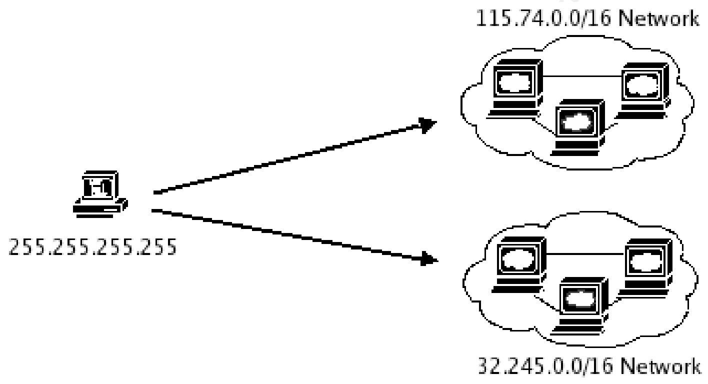
```
[**] [1:184:6] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
10/02-19:03:23.286507 255.255.255.255:31337 -> 115.74.48.253:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]
--
[**] [1:184:6] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
10/02-19:43:56.296507 255.255.255.255:31337 -> 115.74.8.227:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]
--
[**] [1:184:6] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
10/02-20:48:33.286507 255.255.255.255:31337 -> 115.74.76.210:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]
```

Notice the source port 31337, the IP ID of 0 as well as the Sequence, Ack and window size numbers all being 0. Port 31337 "Eleet" is a common port associated with hackers. The presence of the repetitive port 31337 and the repetitive occurrence of the null value in the other packet fields indicate a clear case of packet crafting.

This alert is indicating that an attacker is attempting to open a backdoor with an imbedded command of "cko" on random machines in our home network. However, while investigating the attributes of the Backdoor Q Trojan, and the "cko" payload, I came across some interesting findings. First, I could not identify any occurrences of this actual packet being a direct match to any of the reported

Q Trojan activity. Second, I came across an interesting article on SonicWALL OS resets <http://www.sonicwall.com/services/pdfs/technotes/SonicOS_TCP_RST.pdf>. It seems that the SonicOS generates “cko” and “cki” RST packets for connection cache cleanup. The “cko” payload is indicative of the SonicOS resetting the connection to the responder.

This article is interesting because it most closely matches the actual structure and content of the packets that generated the alert. Without further analysis it would seem that these were in fact a response to spoofed hosts, however it does appear that other tell tale signs of something more are present. The IP ID, for instance, is a clear sign that these packets were crafted rather than generated from the SonicWALL firewall. If the packets had been actual RST packets the IP ID would have incremented.



Backdoor Q can be referenced via CVE number CAN-1999-0660.

<u>Reason detect was selected</u>
Upon my initial analysis of the alert data in ethereal, the 255.255.255.255 source address stood out, and the packets fascinated me. It is not common to see such activity. Due to its most anomalous nature, I felt it an important detect to investigate.

<u>Detect was generated by:</u>
The alert was generated by snort 2.2.0 running on Red Hat Linux kernel version 2.4.20-6. The snort http_inspect preprocessor caught the attack. The trace was also detected while running the raw packets through Ethereal and tcpdump.

The tcpdump command to sort through the capture file is:
#tcpdump -nnqX -r {file} 'host 255.255.255.255'

```
19:03:23.286507 255.255.255.255.31337 > 115.74.48.253.515: tcp 3
0x0000   4500 002b 0000 0000 0f06 9a16 ffff ffff        E..+............
0x0010   734a 30fd 7a69 0203 0000 0000 0000 0000        sJ0.zi..........
0x0020   5014 0000 4f3e 0000 636b 6f00 0000             P...O>..cko...
19:43:56.296507 255.255.255.255.31337 > 115.74.8.227.515: tcp 3
```

```
0x0000   4500 002b 0000 0000 0f06 c230 ffff ffff        E..+.......0....
0x0010   734a 08e3 7a69 0203 0000 0000 0000 0000        sJ..zi..........
0x0020   5014 0000 7758 0000 636b 6f00 0000             P...wX..cko...
20:48:33.286507 255.255.255.255.31337 > 115.74.76.210.515: tcp 3
0x0000   4500 002b 0000 0000 0f06 7e41 ffff ffff        E..+......~A....
0x0010   734a 4cd2 7a69 0203 0000 0000 0000 0000        sJL.zi..........
0x0020   5014 0000 3369 0000 636b 6f00 0000             P...3i..cko...
```

The alert generated was triggered by this snort rule:

```
alert tcp 255.255.255.0/24 any -> $HOME_NET any (msg:"BACKDOOR Q access"; dsize:>1; flags:A+;
flow:stateless; reference:arachnids,203; classtype:misc-activity; sid:184; rev:6;)
```

In this rule we see the source address of 255.255.255.0/24 which means any host in the 255.255.255.X range, which our packets match. Second it is looking for a payload size greater than one byte, which we have a well with the "cko…" (0x636b6f) which is three bytes with three bytes of trailing data.

Probability the source address was spoofed:

High. The sources of the packets are the broadcast address 255.255.255.255. This packet was not sent with the hope of establishing a TCP session.

Attack Mechanism:

Originally thought of as a backdoor attempt, this can also be a part of a DoS attempt for hosts that rely on SonicWall firewalls, or a response to such activity.

Correlations:

While researching these strange series of packets, I came across an article by searching the string "cko rst" by SonicWall firewalls. The article explains that their firewalls will send TCP resets to a responder host if certain conditions were met. In addition to learning that this payload may be attributed to SonicWall, I could not find any other supporting documentation that Q backdoor had ever used this particular form of attack.

Evidence of active targeting:

The packets seem to be targeting random hosts on our internal network. All packets are however structured the same so scanning for a particular backdoor cannot be ruled out.

Severity:

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality = 2

Because this activity is unknown and anomalous in nature, it should be investigated. For this reason I have assigned a value of 2.

Lethality = 3
The objective of the packets is unknown. If this is an attempt to open a backdoor on any of the targeted hosts, then the host would already have to be compromised. For this reason, a value of 3 is appropriate.

System Countermeasures = 1
There are no normal sessions that can be established with such activity, however it is not known if the systems targeted are running any software that would alert on Trojan activity.

Network Countermeasures = 2
Our IDS did pick up this strange activity, however it seems that the broadcast addresses are not filtered before the IDS. It is unknown if any network controls are in place to filter these packets after the IDS. None of the hosts were recorded responding to these packets, but we cannot be sure unless session or full content data is recorded.

Severity = (2 + 3) – (1 + 2) = 2

Detect # 3 – Formmail Exploit

Description of the detect
This detect was analyzed from http://isc.sans.org/logs/raw/2002.9.2, 2002.9.3, 2002.9.9, and 2002.9.10 snort alert logs. I reviewed several days' worth of logs to get a better understanding of what alerts were unique and what alerts were daily and ongoing.

```
[**] [1:1610:5] WEB-CGI formmail arbitrary command execution attempt [**]
[Classification: Web Application Attack] [Priority: 1]
10/02-19:45:08.176507 64.50.23.162:52980 -> 115.74.249.202:80
TCP TTL:110 TOS:0x0 ID:15408 IpLen:20 DgmLen:619 DF
***AP*** Seq: 0x11872EF9  Ack: 0xEE2ABEFE  Win: 0x1800  TcpLen: 20
[Xref => arachnids 226][Xref => cve CVE-1999-0172][Xref => bugtraq 1187][Xref => nessus 10076][Xref =>
nessus 10782]
```

Above is the POST packet used to verify the existence of the formmail.pl file on the web server and determine environmental variables.

```
[**] [1:884:8] WEB-CGI formmail access [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/03-03:45:08.716507 208.187.195.123:50094 -> 115.74.249.202:80
TCP TTL:46 TOS:0x0 ID:32721 IpLen:20 DgmLen:369 DF
***AP*** Seq: 0xB2750819  Ack: 0x2BF2668  Win: 0x60F4  TcpLen: 20

[**] [1:884:8] WEB-CGI formmail access [**]
```

```
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/09-11:38:29.956507 200.30.148.202:4356 -> 32.245.166.119:80
TCP TTL:108 TOS:0x0 ID:16497 IpLen:20 DgmLen:323 DF
***AP*** Seq: 0xAADA5F9  Ack: 0x71A82244  Win: 0x2238  TcpLen: 20
```

These are actual "GET" packets sent to the web server for spam relay, and below is an example of the packet content of the alert above.

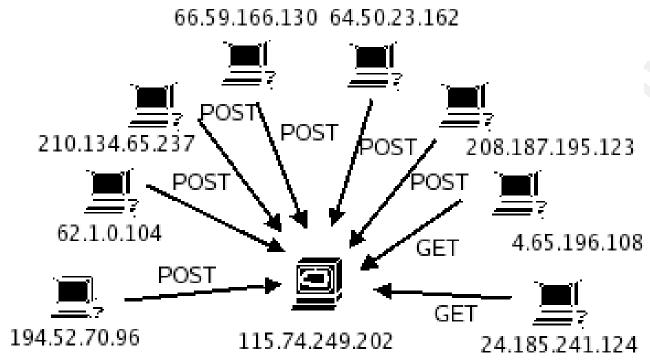
```
[**] WEB-CGI formmail access [**]
10/09-11:38:29.956507 200.30.148.202:4356 -> 32.245.166.119:80
TCP TTL:108 TOS:0x0 ID:16497 IpLen:20 DgmLen:323 DF
***AP*** Seq: 0xAADA5F9  Ack: 0x71A82244  Win: 0x2238  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 01 43 40 71 40 00 6C 06 93 D6 C8 1E 94 CA 20 F5  .C@q@.l....... .
0x0020: A6 77 11 04 00 50 0A AD A5 F9 71 A8 22 44 50 18  .w...P....q."DP.
0x0030: 22 38 6B C4 00 00 47 45 54 20 2F 63 67 69 2D 62  "8k...GET /cgi-b
0x0040: 69 6E 2F 46 6F 72 6D 4D 61 69 6C 2E 63 67 69 3F  in/FormMail.cgi?
0x0050: 65 6D 61 69 6C 3D 72 6F 63 6B 73 74 61 72 40 6D  email=rockstar@m
0x0060: 61 69 6C 2E 63 6F 6D 26 73 75 62 6A 65 63 74 3D  ail.com&subject=
0x0070: 77 77 77 2E 58 58 58 58 58 58 58 58 2F 63 67 69  www.XXXXXXXX/cgi
0x0080: 2D 62 69 6E 2F 46 6F 72 6D 4D 61 69 6C 2E 63 67  -bin/FormMail.cg
0x0090: 69 26 6D 65 73 73 61 67 65 3D 72 6F 63 6B 73 74  i&message=rockst
0x00A0: 61 72 26 72 65 63 69 70 69 65 6E 74 3D 73 6F 6C  ar&recipient=sol
0x00B0: 69 64 73 74 69 68 6C 40 61 6F 6C 2E 63 6F 6D 20  idstihl@aol.com
0x00C0: 48 54 54 50 2F 31 2E 30 0D 0A 56 69 61 3A 20 31  HTTP/1.0..Via: 1
0x00D0: 2E 30 20 43 41 52 45 2D 4E 54 2D 30 31 0D 0A 43  .0 CARE-NT-01..C
0x00E0: 6F 6E 6E 65 63 74 69 6F 6E 3A 20 4B 65 65 70 2D  onnection: Keep-
0x00F0: 41 6C 69 76 65 0D 0A 55 73 65 72 2D 41 67 65 6E  Alive..User-Agen
0x0100: 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 34 2E 30 20 28  t: Mozilla/4.0 (
0x0110: 63 6F 6D 70 61 74 69 62 6C 65 3B 20 4D 53 49 45  compatible; MSIE
0x0120: 20 36 2E 30 3B 20 57 69 6E 64 6F 77 73 20 4E 54   6.0; Windows NT
0x0130: 20 35 2E 31 29 0D 0A 48 6F 73 74 3A 20 6D 61 69   5.1)..Host: mai
0x0140: 6C 2E 63 61 72 65 2E 6F 72 67 2E 67 74 0D 0A 0D  l.care.org.gt...
0x0150: 0A                                               .

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The two targeted IP's were 115.74.249.202 and 32.245.166.119. As stated above, the 32.245.166.119 host is a *nix server running Apache 1.3.12. To make sure that the 115.74.249.202 host was the same host, I analyzed alerts from the 2002.9.2 file to see if any packets contained this IP as the source address. Identical 403 Forbidden alerts were observed:

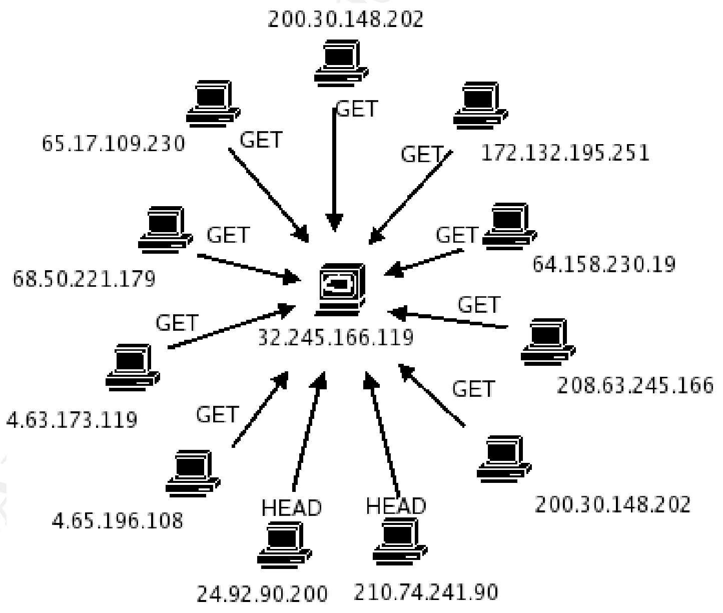
```
11:31:18.036507 115.74.249.202.80 > 213.191.138.153.1697: P [bad tcp cksum 6f6f!]
2765902940:2765903476(536) ack 3917771 win 32696 (DF) (ttl 63, id 23437, len 576, bad cksum a14d!)
0x0000   4500 0240 5b8d 4000 3f06 a14d 734a f9ca        E..@[.@.?..MsJ..
0x0010   d5bf 8a99 0050 06a1 a4dc 545c 003b c7cb        .....P....T\.;..
0x0020   5018 7fb8 9253 0000 4854 5450 2f31 2e31        P....S..HTTP/1.1
0x0030   2034 3033 2046 6f72 6269 6464 656e 0d0a         .403.Forbidden..
0x0040   4461 7465 3a20 5765 642c 2030 3220 4f63         Date:.Wed,.02.Oc
0x0050   7420 3230 3032 2032 303a 3231 3a31 3520          t.2002.20:21:15.
0x0060   474d 540d 0a53 6572 7665 723a 2041 7061          GMT..Server:.Apa
0x0070   6368 652f 312e 332e 3132 2028 556e 6978         che/1.3.12.(Unix
0x0080   2920 2028 5265 6420 4861 742f 4c69 6e75         )..(Red.Hat/Linu
0x0090   7829 2046 726f 6e74 5061 6765 2f34 2e30        x).FrontPage/4.0
0x00a0   2e34 2e33 0d0a 4b65 6570 2d41 6c69 7665          .4.3..Keep-Alive
0x00b0   3a20 7469 6d65 6f75 743d 3135 2c20 6d61         :.timeout=15,.ma
.....<rest of packet omitted>
```

The attacking host sends a crafted HTTP “Get” to the vulnerable formmail script. This is used as a relay to send spam from the vulnerable Web server. The original IP is hidden from the recipient of the spam. It looks as if the exploited Web server is the source.

The Formmail exploit can be referenced via CVE number CAN-2001-0357 and CVE-2000-0411.



Formmail



Formmail2

<u>Reason detect was selected</u>

Keeping in mind that a large majority of alert data was web-related, I chose to do my final analysis on an exploit of a perl script that was used to relay spam anonymously. Being in charge of a sight that is publicly accessible, it is the responsibility of the system and security administrators to maintain the integrity of their sight. The formmail alerts stood out to me because of the fact that so many different hosts were accessing this script. Once initial research was done on the exploitable script, I knew, given the timeframe of the alerts (i.e. 2002) that this was a serious concern.

Detect was generated by:
The alert was generated by snort 2.2.0 running on Red Hat Linux kernel version 2.4.20-6. The snort http_inspect preprocessor caught the attack. The binary alert file was also run through snort 1.9.0 to cross check any alert results. Much like Code Red alerts, snort 2.2.0 flagged these packets with the http_inspect preprocessor. Snort 1.9.0 flagged the formmail as two different alerts shown below. The trace was also detected while running the raw packets through Ethereal and tcpdump.

The snort 1.9.0 rules that triggered the alerts were:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-CGI formmail arbitrary command execution attempt"; flow:to_server,established; uricontent:"/formmail"; nocase; content:"%0a"; nocase; reference:nessus,10782; reference:nessus,10076; reference:bugtraq,1187; reference:cve,CVE-1999-0172; reference:arachnids,226; classtype:web-application-attack; sid:1610; rev:5;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-CGI formmail access"; flow:to_server,established; uricontent:"/formmail"; nocase; reference:nessus,10782; reference:nessus,10076; reference:bugtraq,1187; reference:cve,CVE-1999-0172; reference:arachnids,226; classtype:web-application-activity; sid:884; rev:8;)
```

Both rules are looking for the "formmail" string in the URL, which all the alerts contained. In addition the first rule is also matching the "%0a" which is representative of newline characters used to execute a command in a new line.

Probability the source address was spoofed:

Unlikely. An established TCP session needs to be established to exploit this vulnerability. It is possible however that the IP recorded by both the IDS and the Web server logs is a proxied IP.

Attack Mechanism:

The formmail scripts up to and including version 1.9 were vulnerable to a mail forwarding exploit used by spammers for mail relay. The spammer can supply any recipient email address in the crafted URL to forward any message of their choosing. The formmail script has a flaw in that it does not properly validate the referrer field. A malicious spammer can bypass this check in three ways. One way to do so is to omit the referrer field all together in the URL. This allows the attacker to execute the recipient portion of the code and inject an address of her choosing.

Correlations:

It seems that a handful of Formmail POST requests were sent to the 115.74.249.202 host most likely looking for a vulnerable version of formmail. This is one method for spammers to determine if a vulnerable version of Formmail is evident on a victim web server.

Evidence of active targeting:

The offending IP's are actively searching explicitly for an exploitable version of formmail. This attack is most obviously directed towards this script running on a web server.

Severity:

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality = 3
It can be bad business if this script is exploited and your company is accused of being the propagator of annoying spam emails. If a hacker manages to utilize this script, it should be discovered and the exploit mitigated as soon as possible.

Lethality = 2
This exploit does not corrupt data, but it does cause all spam recipients to contact your company with nasty messages of spam complaints, and can consume company bandwidth. It is therefore suitable to assign a value of two to this category.

System Countermeasures = 1
It is not possible to determine what controls are in place if any. If this script is to be used on a production web server, one of the methods for securing the script should be implemented. Since no information regarding the version, or even the existence of the script is discernable, therefore a value of one is assigned.

Network Countermeasures = 3
From the alert traces it is impossible to determine if any type of content filtering is being performed, however a well placed Network based IDS (all hail Snort!) with current signatures caught the misuse attempt. For this reason a value of three is assigned, due to the successful detect of this activity.

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

4 – Network Statistics

A quick look through tcpdstat shows some protocol usage statistics. Seen here we can see over 97% of all traffic associated with the Snort alerts were Web activity (http). All alerts logged in the three days I chose to analyze were TCP based traffic. This is interesting, as no UDP or ICMP traffic triggered any snort alerts on those days. Another interesting thing that tcpdstat picked up is the gnutella P2P traffic. This also triggered snort alerts, but a quick run through tcpdstat and clear signs of possible policy violations are evident.

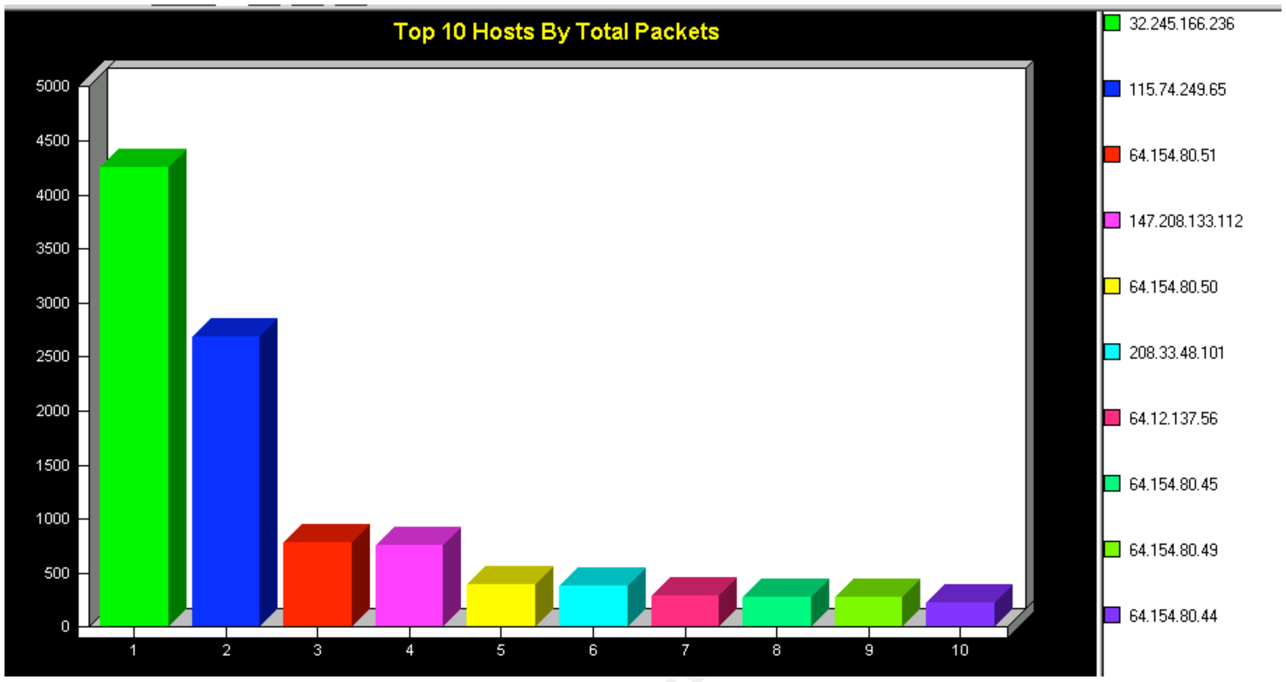
The syntax for this is:

```
$ tcpdstat <cap file>
```

```
.............( Cut )...............
### Protocol Breakdown ###
<<<<
     protocol          packets              bytes          bytes/pkt
------------------------------------------------------------------------
[0] total           7404 (100.00%)         8885397 (100.00%)   1200.08
[1] ip              7404 (100.00%)         8885397 (100.00%)   1200.08
[2]  tcp            7404 (100.00%)         8885397 (100.00%)   1200.08
[3]   ftpdata         53 (  0.72%)           75066 (  0.84%)   1416.34
[3]   ftp              1 (  0.01%)              61 (  0.00%)     61.00
[3]   dns             10 (  0.14%)             600 (  0.01%)     60.00
[3]   http(s)        367 (  4.96%)          415487 (  4.68%)   1132.12
[3]   http(c)       5441 ( 73.49%)         8215667 ( 92.46%)   1509.96
[3]   netb-se          8 (  0.11%)             904 (  0.01%)    113.00
[3]   icecast          3 (  0.04%)            4542 (  0.05%)   1514.00
[3]   gnu6346          9 (  0.12%)            1341 (  0.02%)    149.00
[3]   gnu6347        114 (  1.54%)           12312 (  0.14%)    108.00
[3]   gnu6348         27 (  0.36%)            2916 (  0.03%)    108.00
[3]   gnu6349          5 (  0.07%)             540 (  0.01%)    108.00
[3]   gnu6350         11 (  0.15%)            1188 (  0.01%)    108.00
[3]   gnu6355          1 (  0.01%)             108 (  0.00%)    108.00
[3]   http-a          35 (  0.47%)            2216 (  0.02%)     63.31
[3]   other         1317 ( 17.79%)          152329 (  1.71%)    115.66
[2]  frag             56 (  0.76%)           80148 (  0.90%)   1431.21
```

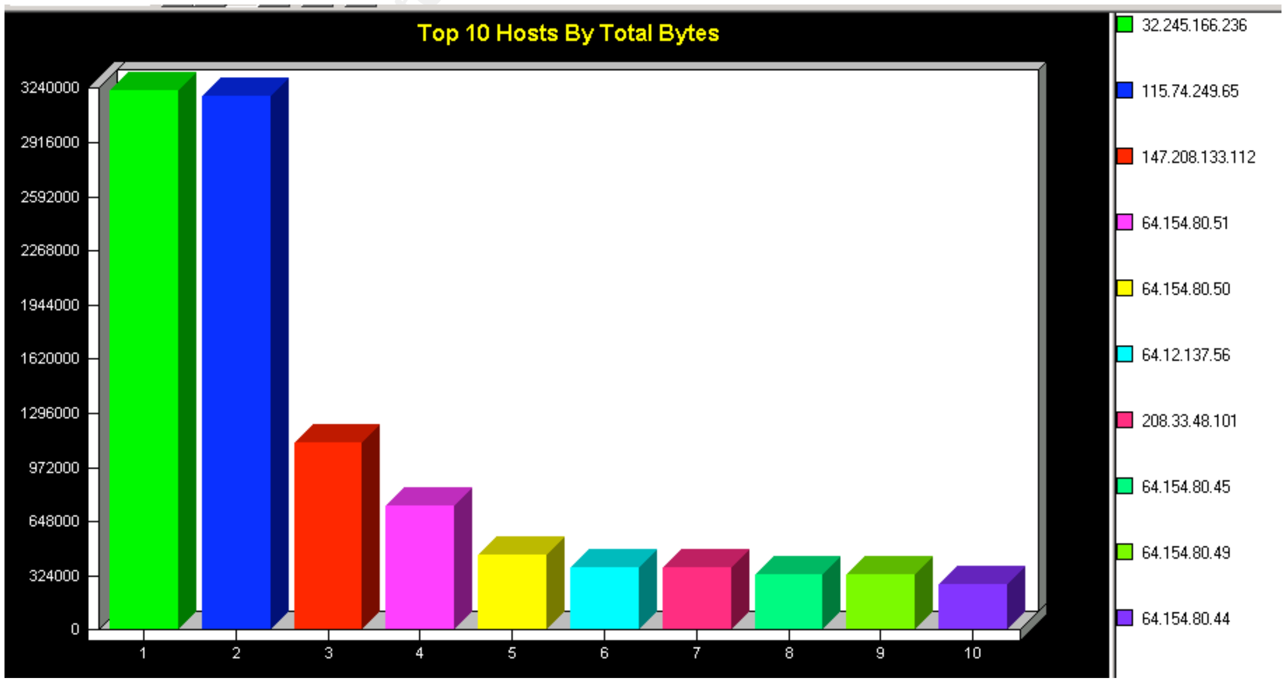
Top talkers:

To determine statistics about top talkers and chattiest hosts, I ran the combined three day alert file through a commercial sniffer. The reports generated show top ten in each category. The categories I filtered on were Top Hosts by Packets, either source or destination. Top hosts in bytes, source or destination and Top talkers by total packets. The results are as follows:



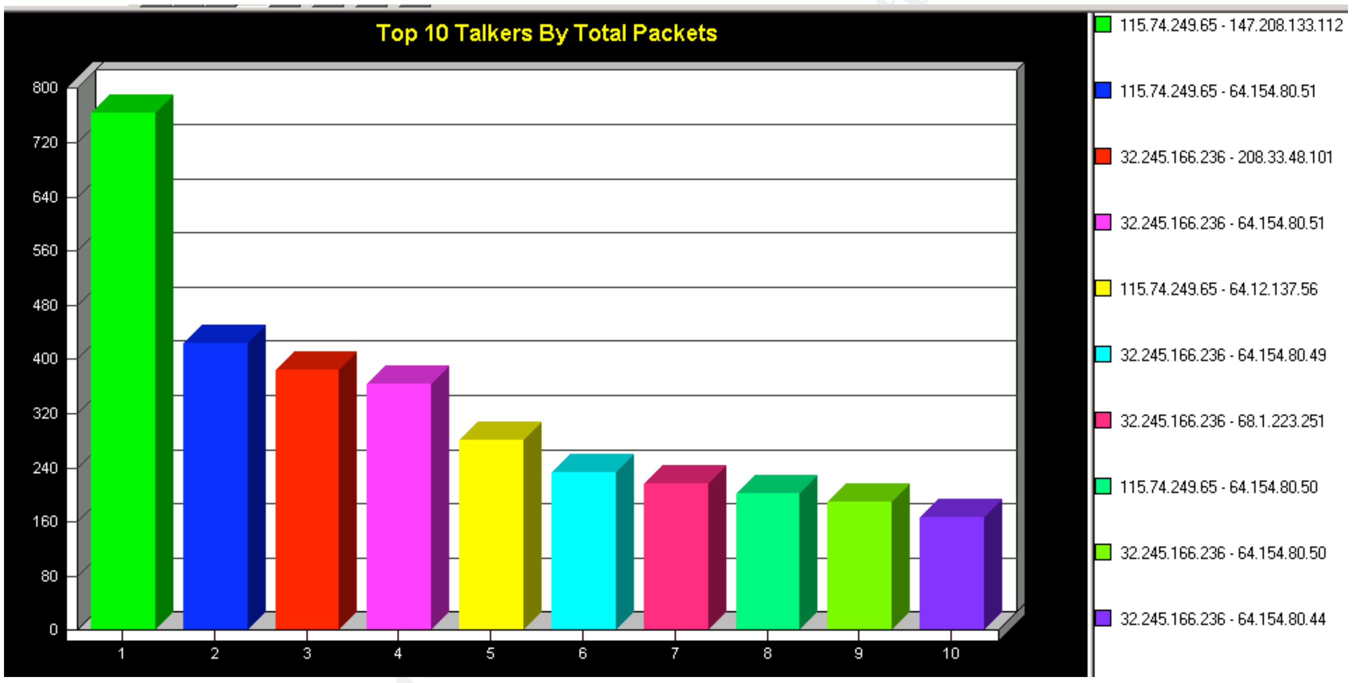
Top Hosts (Total Packets)

#	IP Address	# of Packets	Percentage
1	32.245.166.236	4,255	41.1%
2	115.74.249.65	2,687	25.9%
3	64.154.80.51	788	7.6%
4	147.208.133.112	764	7.4%
5	64.154.80.50	393	3.8%



Top Hosts (total bytes)

#	IP Address	# of Bytes	Percentage
1	32.245.166.236	3,230,906	31%
2	115.74.249.65	3,196,888	30.6%
3	147.208.133.112	1,122,030	10.8%
4	64.154.80.51	744,109	7.1%
5	64.154.80.50	449,777	4.3%



Top Talkers (Total Packets)

#	IP Address	# of Packets	Percentage
1	115.74.249.65 <-> 147.208.133.112	764	23.7%
2	115.74.249.65 <-> 64.154.80.51	424	13.2%
3	32.245.166.236 <-> 208.33.48.101	384	11.9%
4	32.245.166.236 <-> 64.154.80.51	364	11.3%
5	115.74.249.65 <-> 64.12.137.56	280	8.7%

Top 5 targeted services (from above):

The top targeted service from the alerts I reviewed was port 80 (http). This seems to be the most active service that generates alerts for the placement of our IDS.

Top 5 Targeted Services

#	Protocol	Percentage
1	http	78.9%
2	Gnutella	2.3%
3	FTP	.7%
4	DNS	.14%
5	Net Bios	.11%

3 most suspicious hosts:

It is difficult to limit the number of suspicious hosts to three, because any one of the three detects that are covered here have the potential to be devastating to a network. The most suspicious hosts for the detects I chose to write on were mostly Formmail related. The source hosts involved in the Code Red alerts were compromised Web servers, responding to their exploit code. However, the culprit behind the Formmail requests are people who look to utilize vulnerabilities for profit (spammers). It is this activity that worries me most due to the fact that if they are looking for Formmail spam relaying vulnerabilities, they may be looking at other things in the future. True, many spammers run scripts that automate the discovery process and spiders that actually send the spam, but this is a deliberate act. For this reason the hosts involved in the Formmail post requests were the first set of hosts I looked into further. I checked all IP's to determine if they were involved in any future traffic directed to the target network. The addresses 194.52.70.96 and 62.1.0.104 were but two of six address that tried to POST data to the Formmail script. Unfortunately it can be assumed that the spammers use an open proxy to obscure their own IP. It is difficult to determine the OS running on each of these hosts by the ttl (39 and 48 respectively), but my best guess would be some form of Linux OS, due to their proximity to the default ttl of Linux (64). However it is not outside the realm of possibility that these hosts are Windows based, (ttl of 128 for NT or 2000). A traceroute would be beneficial to obtaining a more accurate representation of how far the ttl decremented.

The last external IP I was concerned with was the broadcast address coming into our network with 31337 as its source port. Although not an official host IP, I attempted to sort on other aspects of the packet, such as ttl, however it seems all aspects of the packet are crafted, and untraceable to the real source of the activity. What worries me about this trace is that the router/firewall that is directly in front of the IDS does not filter for broadcast traffic. Also since this activity is so noisy and overtly odd, it could be a form of misdirection.

5 – Correlations from previous GCIA Practicals:

Correlations from previous practicals is a requirement for this version of the GCIA, however I found myself using previous practicals off the bat, because of the work and insight that many students put into their papers. This encompassed not only the analysis on the specific types of packets I chose, but the methodology with which they analyzed their detects regardless of which detects they chose. In searching the Internet I found many links to snippets of practical assignments done by GCIA students. The practicals that stood out in my research were Greg Bassett's paper outlining the Q Trojan, as well as Pete Storm's research done on the strange packets. These papers were very insightful and informative. They gave me a groundwork for starting my analysis for the popular packets.

Gregg Bassett mentioned that he had noticed a small TCP session that was established after the flood of "cko" packets. With this in mind I searched through the traces with ethereal and tcpdump, but no results for suspicious connections were gathered. The conclusions they came up with were solid, and very plausible, however after discovering the SonicWall article, and reading the article at www.sans.org/resources/idfaq/qtrojan.php I was unconvinced that the Q Trojan was the most suitable explanation for this packet. I do believe that it was a crafted packet of some sort, but its origins may be linked to the SonicWall resets, or some failed attempt to exploit them.

As for analytical insight, I drew much inspiration from Jorge Perez's analysis of the shell code NOOP alerts. His in-depth detail into packet header analysis prompted me to look even harder into my own detects, as well as in the workplace.

6 – Insights

The main web server is an Apache 1.3.12 running on Linux as is observable from traffic originating from this host, both 32.245.166.119 and 115.74.249.202. As much as I can tell these two IP's are the same server, knowing that the IP addresses for these alerts have been changed. Being as such, the production web server is not vulnerable to Code Red, however it most likely is running a vulnerable version of Formmail, due to the fact that these traces are from 2002 and Formmail was a very popular script in this year.

Checking the output from tcpdump filtering on the 32.245 net and the 115.74 net it seems as if it is possible based on the ttl values being so close to the default 128, that some of the hosts could be Windows-based. If this is the case then Code Red could possibly make its way into the network and infect a user's workstation that is unwittingly running IIS with their install of Windows.

7 – Defensive Recommendations:

For the Code Red detect, I would recommend several preventative measures. Although it did not affect the production Web server, it is possible and probable that other hosts on your network could be running Windows with IIS installed. Make sure all Anti Virus signatures are up to date, and verify all instances of IIS on your network by using a network scanner. Also, ensure all instances of IIS are patched against the Microsoft ISAPI buffer overflow vulnerability, because variants of worms are known to change their signatures while targeting the same vulnerability.

In addition to that, border routers can also scrub for known payloads. I would recommend investing in a scrubbing router that can match the Code Red signature and filter all related packets as well as all new and existing exploits.

For the Backdoor Q detect, block all packets at your firewall with network or broadcast addresses as the source or destination, as well as all reserved IP address blocks (10.0.0.0/24 192.168.0.0/16 and 172.16-31.0.0/16. These packets are not meant to traverse the Internet, and have no place entering your network. All targeted hosts should be inspected for signs of compromise and for a period of time all communications from and to the targeted hosts should be monitored.

In reference to the Formmail detect, ensure that your production web server is not running a vulnerable version of Formmail. If so, either upgrade or hardcode the recipient IP address into the script to prevent malicious spammers from inserting their own.

The review of the traffic also shows some clear signs of P2P traffic that can easily be blocked from the perimeter firewall. Although this was not a detect that was covered above, it is important that common P2P ports be blocked at the perimeter firewall to prevent file sharing, and possibly an avenue for introducing malicious code into your network.

<u>Part III – Analysis Process</u>

The first step in reviewing the alerts from isc.sans.org/logs/raw was to pick a starting point. The logs I chose first were the 9-9-2002 logs. From this point I wanted to get a feel of what traffic was like prior to and after the day I settled on. This meant the earliest day before which was 9-3-2002 and the day after, 9-10-2002.

The first step in my analysis process was to take a quick glance at these binary alert files through ethereal. Before a single alert was pushed back through Snort, I picked out the Code Red signature due to its distinctive buffer overflow. I feel that a quick glance with a protocol analysis tool gets the analyst more familiar with the types of traffic that he/she will be dealing with in an alert setting.
Next I took the three days worth of files and used a tool bundled with ethereal called mergecap to merge the three files into one big file, so it could be manipulated a little easier. Once I had a file that could be pushed through a number of tools in one operation, I chose to send it through my instance of snort

that I had set up on one of my dual boot Linux/Windows2000 laptop. I use snort with a MySQL database and ACID to view alerts (Pretty standard). I have an instance running both on my Linux partition and on my Windows partition. I created a separate database to store alerts from the SANS logs as to not confuse them with alerts gathered previously.

First, the alerts were tested against snort 2.2.0 with all rule sets turned on. The results were a little confusing in that most of the alerts generated for the Code Red and Formmail were picked up by the http_inspect preprocessor, and not the individual rules written to detect them. I found that they were buried in with false positive alerts generated by miscellaneous web traffic. To cross check my results I downloaded and compiled snort 1.9.0 (not a version suitable for production IDS's). Running the alert file through an older version of snort I generated different results. In these alerts I was able to trigger the Formmail and ISAPI specific alerts. In these alerts were the same packets I picked up while looking through ethereal.

Once the specifics were picked out I used tcpdump, tcpslice and ngrep to sort through and narrow in on specifics that were used in this paper. Along with straight packet analysis, I used Google to learn as much about each individual exploit to better interpret the packets that were being analyzed.

To gather statistical information I used tcpdstat for protocol distribution information and Sniffer Portable for host and session information. These tools were invaluable for in depth statistical data. To run the merged pcap file through the Sniffer Portable engine, I had to utilize the editcap utility also bundled with ethereal. This tool allowed me to change the pcap format into a format compatible with Sniffer Portable. To cross check my results, I also set up a test lab, consisting of a FreeBSD host running tcpreplay, and a Windows host running Sniffer Portable connected to a hub. I ran the merged pcap file through tcpreplay by issuing the command:

```
$ tcpreplay –i <interface> –r .5 <capfile>
```

Listening with Sniffer Portable I came up with the same results as when I opened the converted capture file directly into Sniffer.

References:

Bejtlich, Richard. The TAO of Network Security Monitoring. Boston: Addison-Wesley, 2005.

Larson, Eric and Brian Stevens. Web Servers, Security, And Maintenance. New Jersey: Prentice Hall, 2000.

Castro, Elizabeth. HTML For the World Wide Web. California: Peachpit Press, 2000.

Skoudis, Ed. Malware Fighting Malicious Code. New Jersey: Prentice Hall, 2004.

Stevens, Richard. TCP/IP Illustrated Volume 1, Boston: Addison-Wesley, 1994.

“FormMail hall of shame.” Softwolves.pp.se. 12 Nov. 2004 <http://www.softwolves.pp.se/internet/formmail_hall_of_shame/0209>

Gordon, Less. “SANS: Intrusion Detection FAQ What is the Q Trojan?”. 24 Oct. 2004 <http://www.sans.org/resources/idfaq/qtrojan.php>

“WEB-IIS ISAPI .ida attempt”, Snort Signature Database. 2 Oct. 2004 <http://www.snort.org/snort-db/sid.html?sid=1243>

“BACKDOOR Q access”, Snort Signature Database. 20 Nov. 2004 <http://www.snort.org/snort-db/sid.html?sid=184>

“WEB-CGI formmail arbitrary command execution attempt”, Snort Signature Database. 26 Nov. 2004 <http://www.snort.org/snort-db/sid.html?sid=1610>

“WEB-CGI formmail access”, Snort Signature Database. 26 Nov 2004 <http://www.snort.org/snort-db/sid.html?sid=884>

“FormMail Exploit” Linux/Unix Tutorial Site. 14 Dec. 2004 <http://www.ctssn.com/linux/formMailExploit.html>

“SonicWall Tech Note: SonicOS TCP RST Codes Ver. 1.1” SonicWall. 23 Oct. 2004 <http://www.sonicwall.com/services/pdfs/technotes/ SonicOS_TCP_RST.pdf>

Palamar, Michael. “Formmail.pl Can Be Used As An Open Mail Relay”. 24 Nov. 2004 <http://www.securiteam.com/securitynews/Formmail_pl_Can_Be_

Used_As_An_Open_Mail_Relay.html>

"Backdoor Q". McAfee Security HQ. 15 Oct. 2004
<http://vil.nai.com/vil/content/v_100468.htm>

"Trojan-Active-Q-TCP". Whitehats. arachnids – Intrusion Event Database. IDS203. 20 Oct. 2004 <http://www.whitehats.com/info/IDS203>

Perez, Jorge. "SANS GIAC GCIA Practical Version 3.3".
<http://www.giac.org/practical/GCIA/Jorge_Perez_GCIA.pdf>

Bassett, Greg. "Intrusion Detection: An Inside Look".
<http://www.giac.org/practical/GCIA/Greg_Bassett_GCIA.pdf>

Storm, Pete. "GIAC Certified Intrusion Analyst (GCIA) Practical Assignment Version 3.3". <http://www.giac.org/practical/GCIA/Pete_Storm_GCIA.pdf>

United States Computer Emergency Response Team. CERT Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL. 19 July 2001. 29 Sept. 2004 <http://www.cert.org/advisories/CA-2001-19.html>

"Analysis: .ida "Code Red" Worm." eEye Digital Security. 2 Oct 2004
<http://www.eeye.com/html/Research/Advisories/AL20010717.html>

"Code Red requests for /default.ida." ApacheWeek. 14 Oct 2004
<http://www.apacheweek.com/features/codered>