Intrusion Analysis in Depth

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment
Version: 4.0

Josh Berry

August 18th, 2004

Abstract
The purpose of this paper is to demonstrate competency in the field of intrusion analysis. This is accomplished in three stages:

- Summarize the analyzed intrusion attempts and possible defensive recommendations in a few paragraphs
- Select three events to examine in depth from one day's worth of data. Present findings in the GIAC convention
- Provide an overview of the process used by the analyst to produce the results within the report.

The analysis was performed on events that stood out either because of their uniqueness or because of their potential danger. All examined events have recommendations associated with them on how to mitigate the problem in the future.

Part 1: Executive Summary

Intrusion analysis of the data provided for October 18th, 2002 has been completed per your request. Thorough analysis was performed for the entire day's worth of data and three particular types of events have been examined in depth:

- Anomalous fragmented traffic has penetrated the outer router. This traffic is a variation of the Code Red worm meant to slip by IDS systems undetected. See Part 2, Detect 1.
- An external system is probing your network for the Q Backdoor. This traffic originates from a limited broadcast address (255.255.255.255). See Part 2, Detect 2.
- An external system is probing several of your machines for proxy services on ports 1080, 3128, and 8080. See Part 2, Detect 3.

Most of the traffic presented in the three outlined intrusion detects can be mitigated with the defensive recommendations provided below. Many attacks were directed at the network within this time frame, but no malicious responses were seen coming from within the internal network. This suggests that the exploit attempts were unsuccessful. Other reconnaissance attempts may have been successful in retrieving pre-attack information. Much of the malicious traffic seen directed at your network can be contained or kept to a minimum with some simple recommendations:

- Apply egress filtering to limit outbound connections
- Apply ingress filtering at the router for high-impact ports
- Patch systems regularly to avoid vulnerabilities

Although the network did not appear to be compromised during this time period, immediate attention should be given to the defensive recommendations presented in the report to eliminate several avenues of attack should a vulnerability ever occur within the network.

Part 2: Network Detects

1) Analyzed Scenario

This is the log file used for analysis: http://isc.ssans.org/logs/Raw/2002.10.18
Although the log name date indicates that it was taken on October 18th of 2002, running tcpdump –nnqr 2002.10.18 –tttt | cut –d ‘ ‘ –f 1-2 shows:

-nn – disable name resolution for ports and IP addresses
-q – quiet output (limit protocol information that is displayed)
-r – read 2002.10.18 tcpdump file
-tttt – print timestamp

```
Start Time = 11/18/2002 00:00:41.296507
Stop Time = 11/18/2002 13:45:48.656507
```

These traces were taken with snort running in binary logging mode, only logging packets that matched snort rules as stated in the README file at: http://isc.sans.org/logs/README

2) Network Relationships

2.1 Network Topology

In order to properly estimate the network topology I first needed to know the source and destination MAC's that were captured in the snort binary. To do this I used many of the commands in Andre Cormier's[1] practical:

```
tcpdump –neqr 2002.10.18 | cut –d ‘ ‘ –f 2 | sort | uniq

00:00:0c:04:b2:33
00:03:e3:d9:26:c0
```

And then captured the destination MAC's with this:

```
tcpdump –neqr 2002.20.18 | cut –d ‘ ‘ –f 4 | sort | uniq

00:00:0c:04:b2:33
00:03:e3:d9:26:c0
```

These two MAC addresses were the only sources and destinations for traffic logged by snort. It appears as though the snort sensor is placed in between two network devices. This was confirmed by looking up the MAC's on http://www.ieee.org/web/search[2], which lists the MAC's as assigned to Cisco Systems, Inc.

```
Cisco Device   -------|-------   Cisco Device
                      |
                 Snort Sensor
```

I now need to determine the internal network range. To find the internal network range I produced a list of source IP addresses for the source MAC of 00:00:0c:04:b2:33 (these are slightly different than Andre's scripts; it appears as though the format that is printed has changed in newer versions):

```
tcpdump –neqr 2002.10.18 'ether src 00:00:0c:04:b2:33' | cut –d ' ' –f 9 |
        cut –d . –f 1-4 | sort | uniq

170.129.50.120
170.129.50.3
```

Then destination IP addresses for the source MAC of 00:00:0c:04:b2:33:

```
tcpdump –neqr 2002.10.18 'ether src 00:00:0c:04:b2:33 | cut –d ' ' –f 11 |
        cut –d . –f 1-4 | sort | uniq

144.9.72.134
164.109.22.53
194.67.23.251
194.67.35.196
195.161.116.65
---- [snip] ----
```

Source IP addresses for the source MAC of 00:03:e3:d9:26:c0:

```
tcpdump –neqr 2002.10.18 'ether src 00:03:e3:d9:26:c0' | cut –d ' ' –f 9 |
        cut –d . –f 1-4 | sort | uniq

128.167.120.13
153.33.24.3
161.69.201.238
---- [snip] ----
170.129.15.162
170.129.21.101
---- [snip] ----
```

Destination IP addresses for the source MAC of 00:03:e3:d9:26:c0:

```
tcpdump –neqr 2002.10.18 'ether src 00:03:e3:d9:26:c0' | cut –d ' ' –f 11 |
        cut –d . –f 1-4 | sort | uniq

170.129.100.243
170.129.108.132
170.129.113.233
170.129.113.81
170.129.114.248
---- [ snip] ----
```

After analyzing the data with these commands it appears that the network is designed like this:

```
External ----    Cisco Device    ----------------    Cisco Device      ---- Internal
               00:03:e3:d9:26:c0         |        00:00:0c:04:b2:33       170.129.0.0/16
                                         |
                                    Snort Sensor
```

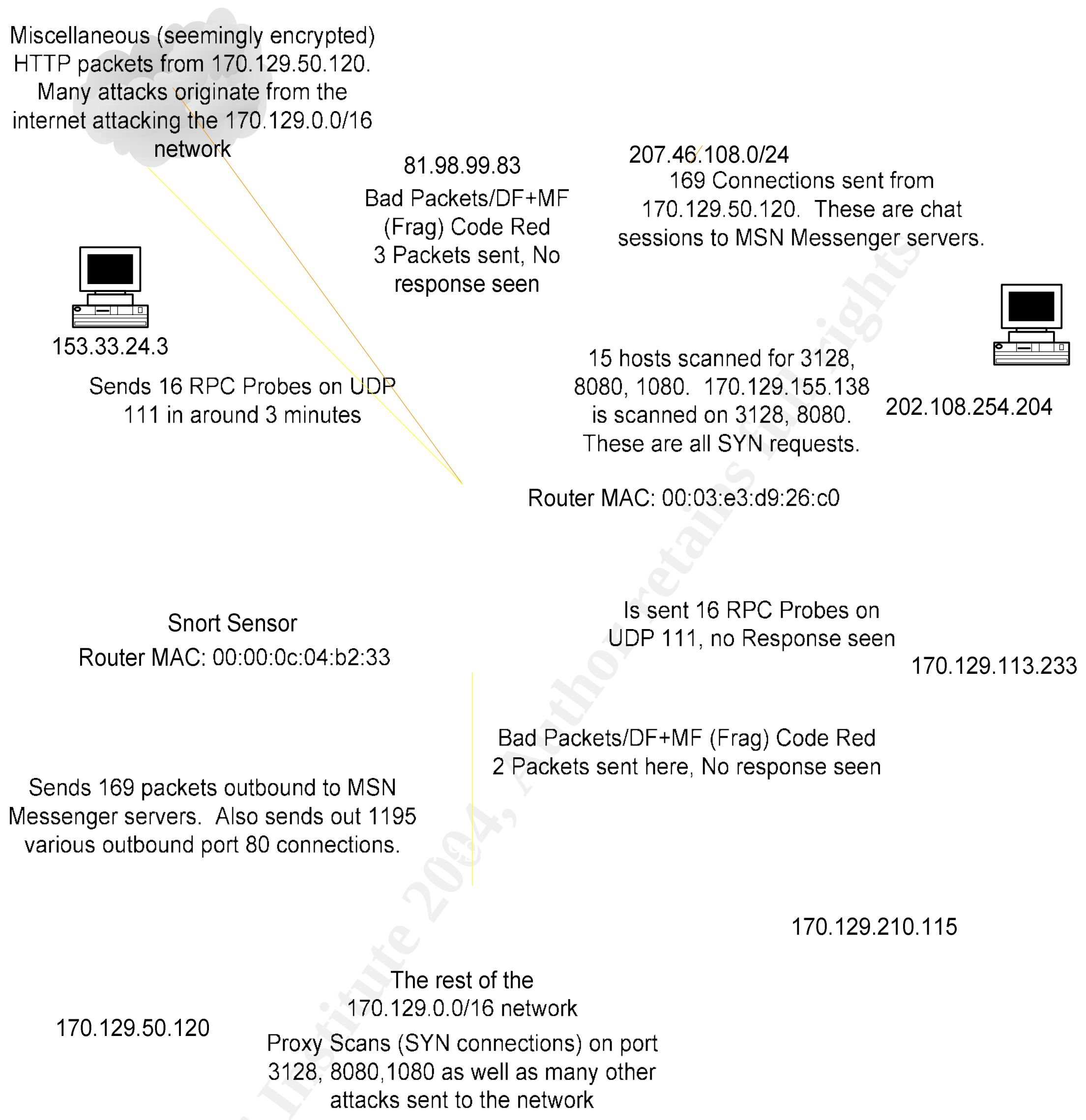
The only peculiarity in this analysis is that some of the 170.129.0.0/16 addresses appear as the source for a source MAC of 00:03:e3:d9:26:c0. Looking at the traffic with:

```
tcpdump –neqr 2002.10.18 ‘ether src 00:03:e3:d9:26:c0 &&
        src net 172.129.0.0/16’
```

```
20:00:02.646507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4
    (0x0800), length 60: IP (tos 0x0, ttl  46, id 0, offset 0, flags [none], length: 28)
    170.129.15.162 > 170.129.15.162: igmp query v2 [gaddr 240.0.3.34]
20:00:02.666507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4
    (0x0800), length 60: IP (tos 0x0, ttl  46, id 0, offset 0, flags [none], length: 28)
    170.129.21.101 > 170.129.21.101: igmp query v2 [gaddr 240.0.1.21]
20:00:02.666507 00:03:e3:d9:26:c0 > 00:00:0c:04:b2:33, ethertype IPv4
    (0x0800), length 60: IP (tos 0x0, ttl  46, id 0, offset 0, flags [none], length: 28)
    170.129.21.133 > 170.129.21.133: igmp query v2 [gaddr 240.0.1.53]
---- [snip] ----
```

These are all IGMP version 2 messages. Tcpdump reports these messages as IGMP queries, however, the group address is set, which is only supposed to occur in IGMP report messages. Another anomaly is the fact that the source and destination IP address are the same. If this were a query, the destination IP address should be set to 224.0.0.1 with a source IP address of the router that sent the query. If this were a report, then the destination address should be the group address with a source IP address of the host[3]. This traffic is either a misconfiguration in the router, a broken client, or spoofed traffic. I believe that this is spoofed traffic because the source MAC is from the external router and the destination address is for the local network, which means that the packet is routable.

2.2 Link Graph


Miscellaneous (seemingly encrypted) HTTP packets from 170.129.50.120. Many attacks originate from the internet attacking the 170.129.0.0/16 network
81.98.99.83
Bad Packets/DF+MF (Frag) Code Red 3 Packets sent, No response seen
207.46.108.0/24
169 Connections sent from 170.129.50.120. These are chat sessions to MSN Messenger servers.
153.33.24.3
Sends 16 RPC Probes on UDP 111 in around 3 minutes
15 hosts scanned for 3128, 8080, 1080. 170.129.155.138 is scanned on 3128, 8080. These are all SYN requests.
202.108.254.204
Router MAC: 00:03:e3:d9:26:c0
Snort Sensor
Router MAC: 00:00:0c:04:b2:33
Is sent 16 RPC Probes on UDP 111, no Response seen
170.129.113.233
Bad Packets/DF+MF (Frag) Code Red 2 Packets sent here, No response seen
Sends 169 packets outbound to MSN Messenger servers. Also sends out 1195 various outbound port 80 connections.
170.129.210.115
The rest of the 170.129.0.0/16 network
170.129.50.120
Proxy Scans (SYN connections) on port 3128, 8080,1080 as well as many other attacks sent to the network

3) Identified Detects

The following statistics show every event that Snort alerted on as well as how many times the event was found and was produced with the following command: egrep '\[**\] \[' 2002.10.18.alerts | sort | uniq -c

Count	GID/SID/REV #	Alert Name
2	[1:1054:6]	WEB-MISC weblogic/tomcat .jsp view source attempt
2	[1:1171:7]	WEB-MISC whisker HEAD with large datagram
2	[1:1201:7]	ATTACK-RESPONSES 403 Forbidden
2	[1:1242:6]	WEB-IIS ISAPI .ida access
2	[1:1243:8]	WEB-IIS ISAPI .ida attempt
3	[1:1288:5]	WEB-FRONTPAGE /_vti_bin/ access

27	[1:1322:6]	BAD-TRAFFIC bad frag bits
2	[1:1610:5]	WEB-CGI formmail arbitrary command execution attempt
21	[1:184:4]	BACKDOOR Q access
3	[1:2381:2]	WEB-MISC schema overflow attempt
8	[1:523:4]	BAD-TRAFFIC ip reserved bit set
16	[1:524:7]	BAD-TRAFFIC tcp port 0 traffic
12	[1:527:4]	BAD-TRAFFIC same SRC/DST
169	[1:540:9]	CHAT MSN message
16	[1:579:7]	RPC portmap mountd request UDP
15	[1:615:5]	SCAN SOCKS Proxy attempt
16	[1:618:5]	SCAN Squid Proxy attempt
16	[1:620:6]	SCAN Proxy Port 8080 attempt
51	[1:628:3]	SCAN nmap TCP
2	[1:884:8]	WEB-CGI formmail access
1	[1:937:6]	WEB-FRONTPAGE _vti_rpc access
1	[1:962:6]	WEB-FRONTPAGE shtml.exe access
4	[1:972:7]	WEB-IIS %2E-asp access
1	[1:990:5]	WEB-IIS _vti_inf access

3.1) Detect 1

3.1.1) Description of Detect 1

The Bad-Traffic Frag Bits signature is used to identify packets that have the Don't Fragment and More Fragments bit set at the same time. This condition is an anomaly because the bits contradict each other; the packet can't be set to not fragment and also be a fragment itself. Here are some snips of the traffic:

Command used for generating this output:

```
tcpdump –vvvttttnnexXSs 65535 –r 2002.10.18 'ip[6] & 32 != 0'
```

vvv – very verbose output
e – display link-layer information
x – display hex information
X – display ASCII information
S – display absolute TCP Sequence number
s 65535 – set the snaplen to 65535
'ip[6] & 32 != 0' – only look at fragmented traffic

```
11/18/2002 09:07:20.516507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 1482:
81.98.99.83.3746 > 170.129.210.115.80: P [bad tcp cksum 3fe3!]
2488519964:2488521392(1428)
 ack 3972886712 win 17520 (frag 14181:1448@0+) (ttl 111, len 1468)
4500 05bc 3765 6000 6f06 7d2c 5162 6353        E...7e`.o.},QbcS
aa81 d273 0ea2 0050 9453 cd1c eccd 70b8        ...s...P.S....p.
5018 4470 cb69 0000 4745 5420 2f64 6566        P.Dp.i..GET./def
6175 6c74 2e69 6461 3f4e 4e4e 4e4e 4e4e        ault.ida?NNNNNNN
```

```
4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e       NNNNNNNNNNNNNNNN
4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e       NNNNNNNNNNNNNNNN
4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e       NNNNNNNNNNNNNNNN
4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e       NNNNNNNNNNNNNNNN
4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e       NNNNNNNNNNNNNNNN
4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e       NNNNNNNNNNNNNNNN
4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e       NNNNNNNNNNNNNNNN
4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e       NNNNNNNNNNNNNNNN
4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e       NNNNNNNNNNNNNNNN
4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e       NNNNNNNNNNNNNNNN
4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e       NNNNNNNNNNNNNNNN
4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e       NNNNNNNNNNNNNNNN
4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e       NNNNNNNNNNNNNNNN
4e4e 4e4e 4e4e 4e4e 4e25 7539 3039 3025       NNNNNNNNN%u9090%
7536 3835 3825 7563 6264 3325 7537 3830       u6858%ucbd3%u780
3125 7539 3039 3025 7536 3835 3825 7563       1%u9090%u6858%uc
6264 3325 7537 3830 3125 7539 3039 3025       bd3%u7801%u9090%
7536 3835 3825 7563 6264 3325 7537 3830       u6858%ucbd3%u780
3125 7539 3039 3025 7539 3039 3025 7538       1%u9090%u9090%u8
3139 3025 7530 3063 3325 7530 3030 3325       190%u00c3%u0003%
7538 6230 3025 7535 3331 6225 7535 3366       u8b00%u531b%u53f
6625 7530 3037 3825 7530 3030 3025 7530       f%u0078%u0000%u0
303d 6120 2048 5454 502f 312e 300d 0a43       0=a..HTTP/1.0..C
6f6e 7465 6e74 2d74 7970 653a 2074 6578       ontent-type:.tex
742f 786d 6c0a 484f 5354 3a77 7777 2e77       t/xml.HOST:www.w
6f72 6d2e 636f 6d0a 2041 6363 6570 743a       orm.com..Accept:
---- [snip] ----
```

and:

```
11/18/2002 09:07:23.086507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 1482:
81.98.99.83.3746 > 170.129.210.115.80: . [bad tcp cksum 474d!]
2488521424:2488522852(1428)
 ack 3972886712 win 17520 (frag 14297:1448@0+) (ttl 111, len 1468)
4500 05bc 37d9 6000 6f06 7cb8 5162 6353       E...7.`.o.|.QbcS
aa81 d273 0ea2 0050 9453 d2d0 eccd 70b8       ...s...P.S....p.
5010 4470 4032 0000 feff ff69 d28d 66f0       P.Dp@2.....i..f.
5089 9574 feff ff8b 4508 8b8d 50fe ffff       P..t....E...P...
8948 108b f48d 952c feff ff52 6a00 8d85       .H.....,...Rj...
4cfe ffff 508d 8dd0 feff ff51 6a00 6a00       L...P......Qj.j.
ff95 98fe ffff 3bf4 9043 4b43 4be9 9f01       ......;..CKCK...
---- [snip] ----
```

Looking further into this packet we see the string GET /default.ida?NNNNNN…. This string is part of the CodeRed exploit. Snort alerted on it for another IP address with this signature:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-IIS ISAPI .ida attempt"; flow:to_server,established; uricontent:".ida?";
nocase; reference:arachnids,552; classtype:web-application-attack;
reference:bugtraq,1065; reference:cve,CAN-2000-0071; sid:1243; rev:8;)
```

and this one:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-IIS ISAPI .ida access"; uricontent:".ida"; nocase;
flow:to_server,established; reference:arachnids,552; classtype:web-application-
activity; reference:cve,CAN-2000-0071; reference:bugtraq,1065; sid:1242;
rev:6;)
```

The fragment anomaly of Don't Fragment and More Fragments bits being set is either caused by a network configuration problem or more probably the worm does this in order to potentially bypass access controls for systems that do not track the state of a connection or to evade intrusion detection systems (it successfully eluding being identified as a CodeRed exploit)[4]. The CodeRed worm in the fragmented packet attempts a buffer overflow of Microsoft's IIS server, infecting the server and propagating itself further. The packet has the PUSH flag set to push the data to the application because more fragments are following. Two more packets are sent from the attacker, one 3 seconds later to the same target and then another one 36 minutes and ten seconds later to another target (170.129.217.170) with this data in common (ASCII periods have been removed):

```
PtEPH,RjLPQjj;CKCK;CKCKLLLLtghm;CKCKL4LLHhPPPh9PsP:LMTHuPLAHRj
---- [snip] ----
```

It appears as though the worm randomly sent the latter part of its payload to 170.129.217.170 or the sensor just missed the first packet, the latter being more probable. Also, each of these fragment alerts all have different fragment ID's verified with this:

```
tcpdump -q -r 2002.10.18 'ip[6] & 32 != 0' | sed -e 's/.*frag //' -e 's/:.*//' | sort | uniq
-c
```

and they have all been crafted to appear to be the beginning of the fragment with the 0+ offset with the length of the datagram being 1448 for a total length of 1468, including the header information. Some of the connections have the same SYN (sequence) numbers which could be a sign of packet crafting but is probably just a retransmission.

The description of the vulnerability that CodeRed takes advantage of can be found here:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0500[5]
or on Microsoft's website at:
http://www.microsoft.com/technet/security/bulletin/MS01-033.mspx[6]

3.1.2) Reason the Detect was Selected
This detect was selected because it was a potentially critical event, and a crafted variation of the previous CodeRed worm that bypassed some access controls and slipped by some IDS systems due to its fragmentation anomaly. This particular instance was not alerted on with the CodeRed rules by Snort and in that respect was successful in slipping by. The worm can be very dangerous, especially when combined with evasion capabilities and was therefore selected as the detect.

3.1.3) Detect Generated By
The platform used for the detect is Slackware 9.0.0 running Snort 2.1.3 (Build 27). The tcpdump version is 3.7.2 and libpcap version 0.7.2. The snort rule set is using the latest set of rules as of 07/30/2004, with every rule turned enabled. The command used to generate the alert file was:

```
snort –c /etc/snort/snort.conf –r 2002.10.18.
```

-c to read the configuration file
-r to read the tcpdump file

The snort.conf file is configured to perform full alert output, logging, decoding of the data link layer, verbose raw packet data dumps, and disabling of checksum calculations with:

```
output alert_full: /var/log/snort/2002.10.18.alerts
config logdir: /var/log/snort
config decode_data_link
config dump_payload_verbose
config checksum_mode: none
```

Multiple alerts can be generated from one packet using this configuration option:

```
config event_queue: max_queue 10 log 3 order_events priority
```

The stream preprocessors have been disabled due to the fact that the packet dumps only have malicious packets and not the full 3-way handshake for each connection/session.

The rule that triggered the alert was:

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"BAD-TRAFFIC bad frag bits"; fragbits:MD; sid:1322; classtype:misc-activity; rev:6;)
```

This rule looks for anomalous packets that have both the More Fragments and Don’t Fragment bits set, which is an illegal condition. Alert generated by snort:

```
[**] [1:1322:6] BAD-TRAFFIC bad frag bits [**]
[Classification: Misc activity] [Priority: 3]
11/18-04:43:33.186507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x5CA
```

```
81.98.99.83 -> 170.129.217.170 TCP TTL:111 TOS:0x0 ID:35584 IpLen:20
DgmLen:1468 DF MF
Frag Offset: 0x0000   Frag Size: 0x05A8
```

3.1.4) Probability the Source Address was Spoofed

There is always a probability of spoofing, however, these are TCP connections and as such require the 3-way handshake completion before successful data transfer, making spoofing much more difficult. Spoofing an address is usually only useful if the attacker does not need a response back from the target such as in a DoS type attack. Further examination of the alert shows that this is actually a CodeRed attack (GET /default.ida?NNNNNNN…) attempting to disguise itself or evade IDS systems in fragmented packets. CodeRed is a worm that takes advantage of an IIS 4.0 and 5.0 and did not spoof its IP, address because it needs to complete the 3-way handshake in order to successfully exploit the system and replicate itself.

3.1.5) Attack Mechanism

CodeRed affects Microsoft Index Server 2.0 and the Windows 2000 Indexing service on computers with Microsoft NT 4.0 or Windows 2000 that are also running IIS 4.0 or IIS 5.0 web servers. The worm uses a known vulnerability in the idq.dll file and is exploitable whether the Indexing service is running or not as long as a mapping in IIS exists for the .ida and .idq files.

CodeRed performs a buffer overflow attack on IIS, exploiting the flaw in the idq.dll file, allowing it to replicate its code to the infected system in memory. Once the worm replicates itself it checks for the file c:\Notworm. If the file exists then CodeRed goes into an infinite sleep. If c:\Notworm does not exist and the day of the month is before the 20th the worm attempts to infect other randomly selected targets and defaces the website. In between the 20th and 28th of the month the worm attempted to DoS www.whitehouse.gov. After July 28th, 2001 the worm ceases to propagate and goes into an infinite sleep state[7].

These CodeRed attacks appear to be unsuccessful because no CodeRed or anomalous fragmented packets are seen coming from the attacked systems as displayed by running tcpdump -nnS -r 2002.10.18.dmp 'ip[6] & 32 != 0' | cut -d ' ' -f 2. Had the CodeRed infection been successful then these systems would have begun attempting to locate and exploit other vulnerable machines.

3.1.6) Correlations

This specific fragmentation attack from CodeRed has been documented here:
http://seclists.org/incidents/2001/Jul/0069.html[4]
The CVE candidate for the vulnerability is listed here:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0500[5]
Microsoft lists information about the vulnerability here:
http://www.microsoft.com/technet/security/bulletin/MS01-033.mspx[6]

A full write-up of the CodeRed worm is available on Symantec's site at: http://securityresponse.symantec.com/avcenter/venc/data/codered.worm.html[7]

The IP address was not found to have any records of being attacked or attacking other systems on dshield.org.

3.1.7) Evidence of Active Targeting

CodeRed distributes itself by scanning and exploiting systems randomly. The attacked systems were more than likely randomly selected and thus it is not active targeting.

3.1.8) Severity

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures). Each value can range between 1 and 5, giving a potential score of +8 or -8.

Criticality = 4. This appears to be a publicly accessible web server and is therefore probably important to the owner of the system.

Lethality = 5. CodeRed causes severe network congestion and usually slows down the infected system to a barely usable level. Variations of the worm (this one being one of those) were suspected of sending potentially confidential information back to www.worm.com.

System Countermeasures = 5. The web server did not begin scanning other systems and thus was probably not vulnerable to the attack either because the server was not running one of the vulnerable versions of IIS or was patched against this vulnerability.

Network Countermeasures = 2. The packets were logged by the sensor, however, the packets were allowed through the network even though they contained an illegal combination of fragment flags.

Severity = (4 +5) – (5 + 2) = 2.

3.2) Detect 2

3.2.1) Description of Detect 2

The BACKDOOR Q access signature is used to identify TCP packets with a source IP address of 255.255.255.0/24, the ACK flag set, and greater than 1 byte in the payload. IP address 255.255.255.255 is a limited broadcast address and should never be the source of a packet[8]. This is a backdoor application for Unix/Linux systems. Some snips of the traffic:

Command used for generating this output:
tcpdump –vvvttttnnexXSs 65535 –r 2002.10.18 'ether src 255.255.255.255'

```
11/18/2002 02:47:15.126507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60:
255.255.255.255.31337 > 170.129.66.181.515: R [tcp sum ok] 0:3(3) ack 0 win
0 [RST cko] (ttl 14, id 0, len 43)
0x0000   4500 002b 0000 0000 0e06 bf97 ffff ffff        E..+............
0x0010   aa81 42b5 7a69 0203 0000 0000 0000 0000        ..B.zi..........
0x0020   5014 0000 73bf 0000 636b 6f00 0000             P...s...cko...

11/18/2002 03:05:32.456507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60:
255.255.255.255.31337 > 170.129.4.175.515: R [tcp sum ok] 0:3(3) ack 0 win 0
 [RST cko] (ttl 14, id 0, len 43)
0x0000   4500 002b 0000 0000 0e06 fd9d ffff ffff        E..+............
0x0010   aa81 04af 7a69 0203 0000 0000 0000 0000        ....zi..........
0x0020   5014 0000 b1c5 0000 636b 6f00 0000             P.......cko...
----- snip -----
```

This is a broadcast IP address being transmitted over TCP which is invalid because TCP is a connection oriented protocol[9]. Another odd fact about the packet is that the source port is 31337, which is often used by hackers, hacking tools, and malicious programs because it stands for the slang term eleet. There are several destination addresses for the 255.255.255.255 traffic, all of which have the RST and ACK flags set. The RST flag being set is interesting because there were no packets originating from the targeted addresses going to 255.255.255.255.

Every packet sent has an IP ID of 0, and a TTL value of 14. This low TTL value along with consecutive IP ID's of 0 are more evidence of packet crafting.

3.2.2) Reason the Detect was Selected

This detect was selected because the systems targeted by 255.255.255.255 should be closely monitored. If this truly is the Q Backdoor it encrypts its content and therefore makes it difficult to determine what the backdoor is attempting to do. Also, the commands are sent in the payload of the packet, but these packets only contain 3 bytes of data. Since it is difficult to determine what this traffic is for and because the packet might be encrypted which further obfuscates the intention, these systems should be watched closely.

3.2.3) Detect Generated by

The platform used for the detect is Slackware 9.0.0 running Snort 2.1.3 (Build 27). The tcpdump version is 3.7.2 and libpcap version 0.7.2. The snort rule set is using the latest set of rules as of 07/30/2004, with every rule turned enabled.

The snort configuration is the same as it was for the first detect.

The rule that triggered the alert was:

```
alert tcp 255.255.255.0/24 any -> $HOME_NET any (msg:"BACKDOOR Q
access"; flags:A+; dsize: >1;  stateless; reference:arachnids,203; sid:184;
classtype:misc-activity; rev:4;)
```

This rule looks for TCP traffic with any IP in the class-C address range of 255.255.255.0/24 with the ACK flag set and more than 1 byte in the payload. This is one of the snort alerts:

```
[**] BACKDOOR Q access [**]
11/18-08:35:06.196507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
255.255.255.255:31337 -> 170.129.26.65:515 TCP TTL:14 TOS:0x0 ID:0
IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 2B 00 00 00 00 0E 06 E8 0B FF FF FF FF AA 81  .+..............
0x0020: 1A 41 7A 69 02 03 00 00 00 00 00 00 00 00 50 14  .Azi..........P.
0x0030: 00 00 9C 33 00 00 63 6B 6F 00 00 00              ...3..cko...
```

3.2.4) Probability the Source Address was Spoofed
There is a high probability that the source address is spoofed. 255.255.255.255 is a limited broadcast that should never be the source of a packet and no machine should be set with this as the IP address. The obvious amount of packet crafting with the low TTL values, IP ID's of 0 and source port of 31337 show evidence of packet crafting and the source is probably crafted as well.

3.2.5) Attack Mechanism
This traffic matches the signature for the Q Backdoor in snort and might be a variation of the backdoor. The original backdoor did not set the RST flag so this might have been done to bypass poor ACL's. The q backdoor qs client is used to send one way IP packets to target machines running the qd daemon. These packets can spawn remote shell processes to which a user can connect with the Q client. The qs client can also send single commands or set up redirection servers on the target. The client also allows the user to spoof the source IP address. This particular attack must be another variation because the IP ID's appear to be crafted as well (the #define Q_ID portion of the code was probably set to 0 at compile time). The Q client is used to connect to encrypted or unencrypted remote shell daemons that were activated using the qs client. Information about how the backdoor is used and operated was obtained through Les Gordon's practical:
http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc[10].

3.2.6) Correlations
This particular traffic has been associated with IRC traffic in the past:
http://archives.neohapsis.com/archives/incidents/2001-05/0038.html[11]
http://lists.jammed.com/incidents/2001/05/0039.html[12]

This traffic was also analyzed by Tu Niem for his practical assignment:
http://cert.uni-stuttgart.de/archive/intrusions/2003/02/msg00008.html[13]
and by Les Gordon in depth at:
http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc[14]

This backdoor is defined in the arachNids database at:
http://www.whitehats.com/info/IDS203[15]

3.2.7) Evidence of Active Targeting
This traffic raises flags on numerous levels. The source port of 31337 is a red flag when analyzing traffic, consistent IP ID's of 0 shows that the packets are being crafted, and RST packets with no stimuli raises the visibility of these attacks or probes. This is probably not active targeting because it is so visible and also the fact that each machine is only targeted once. If the backdoor was installed on these systems there would probably be more of this traffic.

3.2.8) Severity
Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures). Each value can range between 1 and 5, giving a potential score of +8 or -8.

Criticality = 2. The systems are probably being scanned and not actively targeted making the target criticality low.

Lethality = 5. The Q backdoor allows the attacker to run commands on the system and gain a remote shell which can be used to attack other systems, steal information from the infected system, or disrupt the attacked system.

System Countermeasures = 5. The targeted systems did not appear to respond to the traffic or receive more traffic from 255.255.255.255 and were probably not infected with the backdoor program.

Network Countermeasures = 2. The packets were logged by the sensor, however, the packets were allowed through the network even though they are TCP packets from a broadcast address.

Severity = (2 + 5) – (5 + 2) = 0.

3.3) Detect 3
3.3.1) Description of Detect 3
The SCAN SOCKS Proxy attempt signature is used to detect potential probes to port 1080 with the SYN flags set. The socks protocol is referenced at http://www.faqs.org/rfcs/rfc1928.html as RFC 1928[16]. This protocol is designed to be an application layer gateway for TCP based applications and is often used as an HTTP gateway/relay. Some snips of the traffic:

Command used for generating this output:
```
tcpdump –vvvttttnnexXSs 65535 –r 2002.10.18 'ether src 255.255.255.255'

11/18/2002 00:43:59.236507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60:
202.108.254.204.53469 > 170.129.149.62.1080: S [tcp sum
ok] 1844151687:1844151687(0) win 1024 (ttl 46, id 52921, len 40)
0x0000   4500 0028 ceb9 0000 2e06 b51d ca6c fecc        E..(.........l..
0x0010   aa81 953e d0dd 0438 6deb 8587 6deb 8587        ...>...8m...m...
0x0020   5002 0400 e6ed 0000 0000 0000 0000             P............

11/18/2002 01:36:23.816507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60:
202.108.254.204.2897 > 170.129.215.53.1080: S [tcp sum o
k] 1196016012:1196016012(0) win 1024 (ttl 46, id 29679, len 40)
0x0000   4500 0028 73ef 0000 2e06 cdf0 ca6c fecc        E..(s........l..
0x0010   aa81 d735 0b51 0438 4749 c18c 4749 c18c        ...5.Q.8GI..GI..
0x0020   5002 0400 3fbd 0000 0000 0000 0000            P...?.........
----- snip -----
```

This traffic does not indicate any type of packet crafting, the IP ID's, SYN ISN numbers, and source ports are all random numbers. This appears to be a plain vanilla scan for proxy services on three ports (3128/Squid, 1080/Socks, 8080/Microsoft & Other Proxies). The scans are timed 10 to 11 minutes between the first port (8080) and the second port (3128), and then 20 to 21 minutes between the second port (3128) and the third port (1080), and then 20 to 21 minutes before another host is scanned. This could either be a slow scan or the attacker is scanning a large amount of hosts (possibly the 170.129.0.0/16 Class B network) and snort is only alerting on the hosts that are active because connections to non-active hosts are not passed by the router.

3.3.2) Reason the Detect was Selected
This detect was selected because it was a broad port scan looking for multiple proxy ports across an entire network. This is a highly visible reconnaissance attempt. This is the only truly broad scan in the data set.

3.3.3) Detect Generated By
The platform used for the detect is Slackware 9.0.0 running Snort 2.1.3 (Build 27). The tcpdump version is 3.7.2 and libpcap version 0.7.2. The snort rule set is using the latest set of rules as of 07/30/2004, with every rule turned enabled.

The snort configuration is the same as it was for the first detect.

The rule that triggered the alert was:
```
alert tcp $EXTERNAL_NET any -> $HOME_NET 1080 (msg:"SCAN SOCKS
Proxy attempt"; stateless; flags:S; reference:url,help.undernet.org/proxyscan/;
classtype:attempted-recon; sid:615; rev:5;)
```

This rule looks for traffic with a destination port of 1080 and the SYN flag set. This is one of the snort alerts:

```
[**] SCAN SOCKS Proxy attempt [**]
11/17-21:28:48.676507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x3C
202.108.254.204:14924 -> 170.129.252.40:1080 TCP TTL:46 TOS:0x0 ID:25248
IpLen:20 DgmLen:40
******S* Seq: 0x277434C2  Ack: 0x277434C2  Win: 0x400  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 28 62 A0 00 00 2E 06 BA 4C CA 6C FE CC AA 81  .(b......L.l....
0x0020: FC 28 3A 4C 04 38 27 74 34 C2 27 74 34 C2 50 02  .(:L.8't4.'t4.P.
0x0030: 04 00 45 0E 00 00 00 00 00 00 00 00              ..E.........
```

3.3.4) Probability the Source Address was Spoofed

The source IP address (202.108.254.204) is scanning for 3 different proxy addresses (ports 3128, 1080, 8080). The time between scanning port 8080 and 3128 is averaging around 10 to 11 minutes, the time between scanning port 3128 and 1080 is around 20 to 21 minutes, and the time between scanning 1080 and scanning the next host is around 20 to 21 minutes. The source port of the host varies and there are no other systems performing proxy scans. This is the only source scanning for open proxies at the time, the timing is fairly regular, and the attacker needs a response to do anything useful which indicates that the source is not spoofed.

3.3.5) Attack Mechanism

This attack is expecting a response to the TCP probes for ports 1080, 8080, and 3128. The attacker is attempting to find machines that respond with a SYN/ACK indicating that the port is open. If the proxy port happens to be open on one of the scanned systems then further malicious activity might be possible. Improperly configured proxies can be used to bypass ACL's (http://www.securityfocus.com/news/296[17]) giving the attacker access to resources behind the proxy. Other possibilities include relaying spam, relaying attacks against other networks, and anonymously surfing the web through the proxy.

3.3.6) Correlations

This proxy scan was analyzed by GCIA candidates at:
http://cert.uni-stuttgart.de/archive/intrusions/2003/06/msg00267.html[18]
http://lists.sans.org/pipermail/intrusions/2004-June/008067.php[19]

The SANS Internet Storm Center shows that these ports are still highly targeted[20]:
http://isc.sans.org/port_details.php?port=1080
http://isc.sans.org/port_details.php?port=3128
http://isc.sans.org/port_details.php?port=8008

The snort web page contains a description of the alert[21]:
http://www.snort.org/snort-db/sid.html?sid=615
http://www.snort.org/snort-db/sid.html?sid=618
http://www.snort.org/snort-db/sid.html?sid=620

There are many CVE entries for vulnerable proxy servers[22]:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0326
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0371
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0239
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0547
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0471

SecurityFocus had an article on Adrian Lamo and hacking proxies:
http://www.securityfocus.com/news/296 (Reference 17)

3.3.7) Evidence of Active Targeting

This scan is targeting multiple systems on multiple known proxy ports at very slow speeds. The slow speed could be due to a low and slow scan, but could also be due to the attacker scanning the full class B of 170.129.0.0/16. The attacker might be scanning the class B network of 170.129.0.0/16 and the active hosts behind the router are generating alerts from snort. At best this is active network targeting.

3.3.8) Severity

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures). Each value can range between 1 and 5, giving a potential score of +8 or -8.

Criticality = 2. The systems are being scanned randomly and no other proxy traffic is seen going into the network outside of the scans which indicates that proxy services are not running on these machines.

Lethality = 3. Improperly configured proxy servers can allow attackers to bypass ACL's, to browse websites anonymously, or to relay malicious traffic through, such as spam.

System Countermeasures = 5. The only proxy traffic seen going to these systems are the scans. If the systems were running proxy services there would probably be other proxy traffic and alerts generated by snort.

Network Countermeasures = 2. The packets were logged by the sensor, however, the packets were allowed through and should be blocked either by router ACL's or by a firewall.

Severity = (2 + 3) – (5 + 2) = 2.

4) Network Statistics

4.1 Top Talkers

The following source IP addresses represented the sources of the most malicious content.

1) 170.129.50.120 was talking the most, much of this being MSN chat traffic. Snort alerted on the MSN traffic, but looking at more traffic from 170.129.50.120 shows web connections outbound with large chunk of garbled data that might be encrypted. All of this traffic is from high source ports > 60000 and is going outbound to multiple systems on port 80. The other IP addresses were chosen because they represented the most snort alerts.

2) 202.108.254.204 scanned a total of 16 IP addresses for proxy services on ports 8080, 3128, and 1080. Snort alerted on these proxy scans.

3) 255.255.255.255 was chosen as a top talker for several reasons. This address is a limited broadcast address transmitting TCP packets (See reference number 8). TCP is a connection oriented protocol and therefore cannot broadcast packets[23]. Also, this address should not be the source of a packet. Snort alerts on these packets as being BACKDOOR Q access. This type of traffic was discussed in several locations [http://www.securityfocus.com/archive/19/187958[24]].

4) 153.33.24.3 triggered RPC Portmap mountd requests. These packets made it past the first layer in the network and could potentially return very useful information to attackers. RPC services should never be accessible from the internet.

5) 211.47.255.23 triggered the BAD-TRAFFIC tcp source port 0 alert from snort. Port 0 is an invalid port number. Looking at the packets sent by this IP address, it appears to send an initial SYN request, followed by 3 more requests with the same ISN, four times within a 2 minute time frame. The following requests are probably retransmissions and this is probably some form of reconnaissance probe.

Count	Source IP Address
1364	170.129.50.120
47	202.108.254.204
21	255.255.255.255
16	153.33.24.3
16	211.47.255.23

4.2 Top Targeted Ports

1) 80 – Port 80 traffic generated 76 snort alerts. These alerts ranged from CodeRed based alerts, to Frontpage alerts, to anomalous fragmented packet

alerts. Port 80 is being actively targeted because it is often accessible from the internet and is usually the port used to run web servers which have been affected by numerous vulnerabilities.

2) 0 – Port 0 is an invalid port number. The associated connection attempts are probably reconnaissance probes.

3) 515 – This port is targeted by 255.255.255.255 with reset connections. Port 515 is the print spooler port and should not be allowed in from outside networks.

4) 111 – This port is usually used by the Portmap service to map RPC services to port numbers. This port should never be accessible from outside networks as it can provide a wealth of valuable reconnaissance information and has been associated with dangerous vulnerabilities in the past[25].

5) 3128 – This is one of 3 proxy ports that were probed from one IP address. The attacker was probably trying to find an open proxy to allow anonymous web browsing or to relay attacks through.

Count	Destination Port
1280	80
21	515
16	0
16	111
16	3128

<u>4.3 Three Most Suspicious Source Addresses</u>

1) 255.255.255.255 - This address is suspicious because it is a broadcast address transmitting TCP connections, which is a protocol violation because TCP is a connection oriented protocol and thus should not broadcast traffic (See reference 21). All traffic with source 255.255.255.255 has a source port of 31337, a common port used by malicious programs that spells eleet. There is no registration information for this address because it is a broadcast address.

2) 153.33.24.3 – This address is targeting the portmap service on 170.129.113.233. Portmapper maps RPC services to port numbers which can be further probed for valuable reconnaissance information. This service should not be accessible from an external address. The probes coming from this address have a source port of 965, indicating that the user has root level privileges and that the user is probably running some form of RPC scanner or exploit that crafts the source port. Registration information for 153.33.24.3[26]:

OrgName:	LTX Corp.
OrgID:	LTXCOR-1
Address:	3930 N. First St.
City:	San Jose

StateProv: CA
PostalCode: 95134
Country: US

NetRange: 153.33.0.0 - 153.33.255.255
CIDR: 153.33.0.0/16
NetName: LTX-SAN-JOSE
NetHandle: NET-153-33-0-0-1
Parent: NET-153-0-0-0-0
NetType: Direct Assignment
NameServer: SJ-DNS1.LTX.COM
NameServer: NS-240A.LTX.COM
Comment:
RegDate: 1992-12-07
Updated: 2003-10-03

TechHandle: TK29-ARIN
TechName: Kemmerling, Todd
TechPhone: +1-408-383-2438
TechEmail: kemmer@ltx.com

OrgTechHandle: DEC10-ARIN
OrgTechName: Christman, David E.
OrgTechPhone: +1-408-383-2420
OrgTechEmail: david_christman@ltx.com

3) 211.47.255.23 – This IP is listed as a suspicious address because the host is sending SYN requests to 170.129.235.40 on port 0. Port 0 is an invalid port number and needs to be blocked by an ACL on the router. Port 0 indicates packet crafting and is probably used for probing to elicit a response for determining whether the machine is connected or what operating system is running on 170.129.235.40. Registration information for 211.47.255.23 (See reference 26):

OrgName: Asia Pacific Network Information Centre
OrgID: APNIC
Address: PO Box 2131
City: Milton
StateProv: QLD
PostalCode: 4064
Country: AU

ReferralServer: whois://whois.apnic.net

NetRange: 210.0.0.0 - 211.255.255.255
CIDR: 210.0.0.0/7

```
NetName:    APNIC-CIDR-BLK2
NetHandle:  NET-210-0-0-0-1
Parent:
NetType:    Allocated to APNIC
NameServer: NS1.APNIC.NET
NameServer: NS3.APNIC.NET
NameServer: NS4.APNIC.NET
NameServer: NS.RIPE.NET
NameServer: TINNIE.ARIN.NET
NameServer: DNS1.TELSTRA.NET
Comment:    This IP address range is not registered in the ARIN database.
Comment:    For details, refer to the APNIC Whois Database via
Comment:    WHOIS.APNIC.NET or http://www.apnic.net/apnic-bin/whois2.pl
Comment:    ** IMPORTANT NOTE: APNIC is the Regional Internet Registry
Comment:    for the Asia Pacific region. APNIC does not operate networks
Comment:    using this IP address range and is not able to investigate
Comment:    spam or abuse reports relating to these addresses. For more
Comment:    help, refer to http://www.apnic.net/info/faq/abuse
Comment:
RegDate:    1996-07-01
Updated:    2004-03-30

OrgTechHandle: AWC12-ARIN
OrgTechName:   APNIC Whois Contact
OrgTechPhone:  +61 7 3858 3100
OrgTechEmail:  search-apnic-not-arin@apnic.net
```

5) Correlations from other Students

5.1 Detect 1

This CodeRed exploit has been documented by several GCIA candidates:
http://cert.uni-stuttgart.de/archive/intrusions/2003/12/msg00176.html[27]
http://cert.uni-stuttgart.de/archive/intrusions/2002/08/msg00106.html[28]
http://cert.uni-stuttgart.de/archive/intrusions/2002/09/msg00407.html[29]

5.2 Detect 2

This backdoor was analyzed by Tu Niem for his practical assignment:
http://www.giac.org/practical/GCIA/Tu_Niem_GCIA.pdf (See reference 13)
and by Les Gordon in depth at:
http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc (See reference 14)
The backdoor was also analyzed with firewall traffic by John Jenkinson at:
http://www.giac.org/practical/GCIA/John_Jenkinson_GCIA.doc[30]

5.3 Detect 3

Bruce Auburn examined Proxy scanning and some specific scan tools at:
http://www.giac.org/practical/GCIA/Bruce_Auburn_GCIA.pdf[31]

Stephen Breault examined these alerts at:
http://lists.sans.org/pipermail/intrusions/2004-June/008067.php (See reference 19)
More GCIA candidates analyzed these alerts here:
http://cert.uni-stuttgart.de/archive/intrusions/2003/06/msg00267.html (See reference 18)

6) Other Insight

There are several types of traffic entering the network that could be blocked by the external router to provide an extra layer of security. Some of this anomalous traffic should be dropped by a router configured and functioning properly. Access lists could block port 0 traffic and port 111 RPC traffic. Also, if configured correctly, the external router should not forward 255.255.255.255 broadcast traffic.

7) Defensive Recommendations

7.1 Detect 1

The best way to defend against this attack is to apply the appropriate patch for IIS, information about where to get the patch can be found here:
http://www.microsoft.com/technet/security/bulletin/MS01-033.mspx (See reference 6), or a cumulative patch fixing this vulnerability and many others in IIS at:
http://www.microsoft.com/technet/security/bulletin/MS01-044.mspx[32]

There should be network based access control devices put in place to keep state of connections and block connections that violate protocol standards. The idq.dll file can also be unmapped from IIS to prevent the system from being vulnerable to future problems with the Index Service.

7.2 Detect 2

Broadcast addresses as sources should be blocked at the perimeter by either the router or a firewall. Also, a firewall that tracks connection state should be deployed to prevent packets with RST and no stimuli from entering the network. Also, the printer port (515) should not be accessible from the outside and can be blocked by the router or with a firewall.

7.3 Detect 3

Proxy servers are generally used to provide a gateway to the internet for internal clients, unless used as reverse proxies. Since the proxy server usually is provided as a layer of protection for web browsers within a network, these ports can be blocked coming inbound from either the external router or from a firewall. If any proxy servers do reside on the internal network, proper configuration needs to be checked to ensure external access is blocked and that user authentication is required.

Part 3: Analysis Process

1) System Setup
Log data for analysis was downloaded onto a Slackware 9.0.0 system. Snort version 2.1.3 (Build 27), Tcpdump 3.7.2, and libpcap 0.7.2 were all installed on the system prior to any analysis.

2) Pre-Analysis Configuration
The snort rule set was updated with oinkmaster, with this configuration option to enable all rules:

```
modifysid * ‘^# alert’ | ‘alert’
modifysid * ‘^#alert’ | ‘alert’
```

After enabling all the rules I removed the state tracking for each rule since the snort binary file only contained malicious connections (not the full 3-way handshakes). In order to accomplish this I created a small perl script to strip out the flow combinations for each rule file:

```
#!/usr/bin/perl -i
my($freg, $sreg);
while(<>) {
        if ($_ =~ /^alert/) {
                $_ =~ s/(.*)\s+flow\:/$1\s/;
                $freg = $1;
                $_ =~ s/\;\s+(.*)/\s$1/;
                $sreg = $1;
                $_ = "$freg $sreg\n";
                print;
        } else {
                print;
        }
}
```

Which was run like this: ./remove_flow.pl <rulefile.rules>. The solution is ugly but it worked. After cleaning up the rules, configurations in the snort.conf file were set. The stream4 preprocessors were disabled to ignore the state of connections. Data was sent to several places, full alert output was sent to a log file and CVS output was enabled for easy data manipulation. Hex packet logs were sent to /var/log/snort:

```
output alert_full: /var/log/snort/2002.10.18.alerts
output alert_CSV: /var/log/snort/2002.10.18.csv
     timestamp,msg,ethsrc,src,srcport,ethdst,dst,dstport
config logdir: /var/log/snort
```

Snort was configured to decode layer 2 traffic and dump raw payload information at a verbose level with these options:

```
config decode_data_link
config dump_payload_verbose
```

The checksums in the snort binary files were changed to disable the possibility of using them to figure out the true IP addresses in the dump file. Snort automatically checks for proper checksums so this was disabled with:

```
config checksum_mode: none
```

In the past, Snort would alert on the first signature that a payload matched. If multiple events were contained in one packet, only one would be alerted on. Recently, in Snort 2.1.3RC1, event queue support was created to alert on multiple events per packet and prioritize these alerts. This configuration option was enabled like this:

```
config event_queue: max_queue 10 log 3 order_events priority
```

<u>3) Data Analysis</u>

Once everything was configured as needed I generated the alert logs by running snort against the binary file with:

```
snort –c /etc/snort/snort-gcia.conf –r 2002.10.18
```

The alert files were parsed in various ways to help data analysis. I found all unique alerts running with this:

```
egrep '\[\*\*\] \[' 2002.10.18.alerts | sort | uniq –c
```

Unique IP sources and unique IP destinations as well as Unique IP source and destination ports:

```
cat 2002.10.18.csv | cut -d , -f 4 | sort | uniq -c
cat 2002.10.18.csv | cut -d , -f 7 | sort | uniq -c
cat 2002.10.18.csv | cut -d , -f 5 | sort | uniq –c
cat 2002.10.18.csv | cut -d , -f 8 | sort | uniq –c
```

Other analysis of the snort binary was done using original and modified commands from Andre Cormier's practical (See reference 1). These were used to gather source and destination MAC's, source and destination IP's from each MAC:

<u>Unique Source MAC's:</u>

```
tcpdump –neqr 2002.10.18 | cut –d ' ' –f 2 | sort | uniq
```

<u>Unique Destination MAC's:</u>

```
tcpdump –neqr 2002.20.18 | cut –d ' ' –f 4 | sort | uniq
```

<u>Unique Source IP's From 00:00:0c:04:b2:33:</u>

```
tcpdump –neqr 2002.10.18 'ether src 00:00:0c:04:b2:33' | cut –d ' ' –f 9 |
        cut –d . –f 1-4 | sort | uniq
```

Unique Destination IP’s From 00:00:0c:04:b2:33:

```
tcpdump –neqr 2002.10.18 ‘ether src 00:00:0c:04:b2:33 | cut –d ‘ ‘ –f 11 |
        cut –d . –f 1-4 | sort | uniq
```

Unique Source IP’s From 00:03:e3:d9:26:c0:

```
tcpdump –neqr 2002.10.18 ‘ether src 00:03:e3:d9:26:c0’ | cut –d ‘ ‘ –f 9 |
        cut –d . –f 1-4 | sort | uniq
```

Unique Destination IP’s From 00:03:e3:d9:26:c0:

```
tcpdump –neqr 2002.10.18 ‘ether src 00:03:e3:d9:26:c0’ | cut –d ‘ ‘ –f 11 |
        cut –d . –f 1-4 | sort | uniq
```

After this analysis was completed I was able to accurately define the network topology and create link graphs associated with various machines. This information was then used to fully analyze the attacks seen on the network. While analyzing the data I realized that although all of the network traffic within the binary file was logged because it set off a preprocessor or signature in snort, when I ran snort against the file not every connection was alerted on. I attribute this lack of alerts to the enhancements that have been developed in snort since the binary file was created.

References:
References have been included within the body of the document.

---

[1] Cormier, Andre. “GCIA Practical.” Feb. 5, 2003.
URL: http://www.giac.org/practical/GCIA/Andre_Cormier_GCIA.pdf

[2] IEEE. "Search." IEEE Home page. Jul 2004
URL: http://www.ieee.org/web/search/

[3] Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Unknown: China Machine Press. 1994. 180-181.

[4] Bejtlich, Richard. “IIS .ida exploit involving worm.com/181.com/ 216.99.52.100.” Jul 15, 2001. URL: http://seclists.org/incidents/2001/Jul/0069.html

[5] Mitre CVE Database. “Buffer overflow in ISAPI extension (idq.dll).”
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0500

[6] Microsoft Security Bulletin. “Unchecked Buffer in Index Server ISAPI Extension Could Enable Web Server Compromise.”
URL: http://www.microsoft.com/technet/security/bulletin/MS01-033.mspx

[7] Symantec Security. “CodeRed Worm.” July 16, 2001.
URL:
http://securityresponse.symantec.com/avcenter/venc/data/codered.worm.html

[8] Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Unknown: China Machine Press. 1994. 171.

[9] Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Unknown: China Machine Press. 1994. 169.

[10] Gordon, Les. “GCIA Practical.” Nov 22, 2002.
URL: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc

[11] Peterson, Jeff. “Backdoor Q access.” May 4, 2001.
URL: http://archives.neohapsis.com/archives/incidents/2001-05/0038.html

[12] Storm, Jason. “Backdoor Q access.” May 4, 2001.
URL: http://lists.jammed.com/incidents/2001/05/0039.html

[13] Niem, Tu. “GCIA Practical.” Jan 23, 2003.
URL: http://www.giac.org/practical/GCIA/Tu_Niem_GCIA.pdf

[14] Gordon, Les. “GCIA Practical.” Nov 22, 2002.
URL: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc

[15] Whitehat ArachNIDS Database. “Trojan-Active-Q-TCP”.
URL: http://www.whitehats.com/info/IDS203

[16] RFC Database. “RFC 1928: SOCKS Protocol.”
URL: http://www.faqs.org/rfcs/rfc1928.html

[17] Poulsen, Kevin. “Lamo's Adventures in WorldCom”. SecurityFocus. Dec 5, 2001. URL: http://www.securityfocus.com/news/296

[18] Wittich, Don, Pat and Dondi. GCIA Practical: Detect 1. Jun 22, 2003.
URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/06/msg00267.html

[19] Breault, Stephen. GCIA Practical: Detect 1. Jun 5, 2004.
URL: http://lists.sans.org/pipermail/intrusions/2004-June/008067.php

[20] SANS Stormcenter. Ports 1080, 8080, 3128. Aug 2004
URL: http://isc.sans.org/port_details.php?port=1080
URL: http://isc.sans.org/port_details.php?port=3128
URL: http://isc.sans.org/port_details.php?port=8080

[21] Snort Database. Scan for Squid/Socks/Other Proxy.
http://www.snort.org/snort-db/sid.html?sid=615
http://www.snort.org/snort-db/sid.html?sid=618
http://www.snort.org/snort-db/sid.html?sid=620

[22] Mitre CVE Database. Vulnerable Proxy Servers.
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0326
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0371
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0239
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0547
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0471

[23] Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Unknown: China Machine Press. 1994. 169, 223.

[24] Horsfall, Dave. “Re: Probe from 255.255.255.255.”. SecurityFocus. Jun 1, 2001. URL: http://www.securityfocus.com/archive/19/187958

[25] SecurityFocus BID Database. RPC Vulnerabilities.
URL: http://www.securityfocus.com/bid/422
URL: http://www.securityfocus.com/bid/1892
URL: http://www.securityfocus.com/bid/3400

[26] ARIN Database. Whois Entries.
URL: http://ws.arin.net/cgi-bin/whois.pl

[27] Williams, Todd. GCIA Practical: Detect 1. Dec 30, 2003.
URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/12/msg00176.html

[28] Gregory, Scott. GCIA Practical: Detect 1. Aug 10, 2002.
URL: http://cert.uni-stuttgart.de/archive/intrusions/2002/08/msg00106.html

[29] McCabe, Mike. GCIA Practical. Nov 3, 2002.
URL: http://www.giac.org/practical/GCIA/Mike_McCabe_GCIA.doc

[30] Jenkinson, John. GCIA Practical. Aug 13, 2001.
URL: http://www.giac.org/practical/GCIA/John_Jenkinson_GCIA.doc

[31] Auburn, Bruce. GCIA Practical. Jul 8, 2003.
URL: http://www.giac.org/practical/GCIA/Bruce_Auburn_GCIA.pdf

[32] Microsoft Security Bulletin. “Cumulative Patch for IIS.” Aug 15, 2001.
URL: http://www.microsoft.com/technet/security/bulletin/MS01-044.mspx