Global Information Assurance Certification Paper

Abstract

Within this paper are 3 sections; Executive Summary, Detailed Analysis, and the Analysis Process. Part 1, The Executive Summary, details the security state of the University and recommendations for improvement. Part 2, Detailed Analysis, includes information on the events found in the traffic dumps and the three critical detects analyzed as per assignment; MISC Tiny Fragments, Backdoor Q Access, and BAD-TRAFFIC ip reserved bit set. Part 3, the Analysis Process, details the methods used to generate the conclusions.

Provided in addition to the 3 sections are a link graph, an inferred network topology, and additional charts detailing the network traffic observed.

Document Conventions

`Computer`	Operating system commands and computer output are represented in this format.
Filenames	File names are represented in this font.
URL's	Weblinks are represented in this format.

GIAC Certified Intrusion Analyst
(GCIA)
Practical Assignment
Version 4.0
The Challenges of Intuitional Security

Kevin G. Holestine
GCIA / Denver CO
July 10 to 15 2004

Part I – Executive Summary

A university's computer network is principal to its ability to conduct business. For that reason the security of the network is as important as the professors that the university employs. To an attacker, a university is a model target. The large bandwidth and high number of users make it ideal for obfuscating their activity. So, to understand how to better defend a network, the items that you must protect need to be identified. For a university, those items are sorted into four general categories.

Intellectual property
Financial records
Student records
Network integrity

The individual users who will need specific accesses within these categories are classified to the following three user groups.

Students
- Comprise the largest amount of traffic being generated at any given time
- Broad in need and access privileges

Academic Staff
- Has higher security requirements and access needs
- Must be able to compensate for the addition of temporary/visiting faculty

Administrative/functional Staff,
- Ensures the proper day to day execution of university business
- Should have specific/well defined access needs

The secure access by these users to the protected information is crucial to the integrity of the University.

Within this report I have analyzed two days worth of data, *2002.9.19* and *2002.9.2*0. Using Snort ® 2.2 intrusion detection software the total alerts logged from the files was 191 out of 20,300 total packets. The alerts were generated by Snort® with the Dec 1st rule set installed and all rules enabled.

I selected 3 detects which I consider the most critical, and analyzed those with more scrutiny.

MISC tiny fragments
Suspicious traffic with evidence of packet crafting for malicious purposes

Backdoor Q
Possible backdoor/Trojan

IP reserved bit set
Suspicious traffic with evidence of packet crafting for malicious purposes

In addition to the critical detects I noted additional suspicious traffic from IPs:

148.63.228.0 - possible password transmission in clear text.
217.228.210.78 - fast and large SYN scan on Home Network.
24.190.48.235 - fast and large SYN scan on Home Network.

There is no evidence to support the existence of compromised systems within the network, although without a full network traffic capture this can not be ruled out as ensuing 2-way traffic may appear to be legitimate to IDS systems. However, there is plenty of evidence indicating active scanning for compromised devices. There is also evidence of scanning in the form of reconnaissance designed for the purpose of OS fingerprinting, which allows an attacker to isolate any future attacks to be specific to the operating systems residing on the destination hosts.

Based on the traffic analyzed my defensive recommendations for this network are:

The security of the network needs to be strengthened at the perimeter.
Specifically:

- Firewall configuration blocking inbound traffic from broadcast addresses.
- Firewall configuration blocking inbound traffic to port 515.
- Firewall configuration blocking inbound traffic with reserved bit set.
- Firewall configuration blocking inbound traffic with source port 0.
- Enforce rules that require the minimum size of the first fragment to be great enough to contain all of the protocol header information.

Ensure all network hosts are patched with the latest vendor security patches.
Specifically:

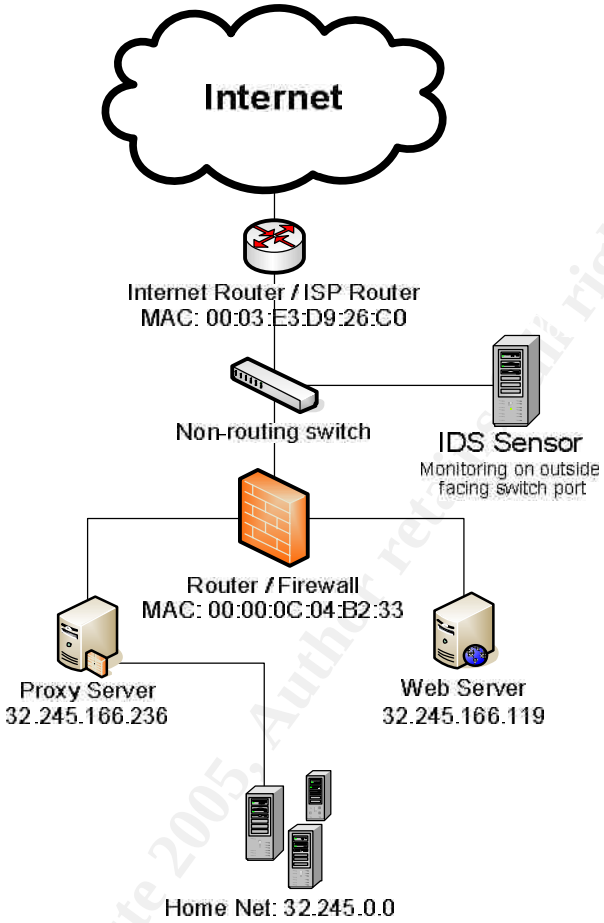
- Institute a diligent patch management process for all publicly available network resources.

Institute strict Anti-virus and personal firewall protection with group policy assignments.
Specifically:

- Ensure that all host's virus definitions are up to date.
- Ensure that a personal firewall application has been installed and is running on all hosts in the network.

Part II – Detailed Analysis

Diagram 1: Inferred network topology

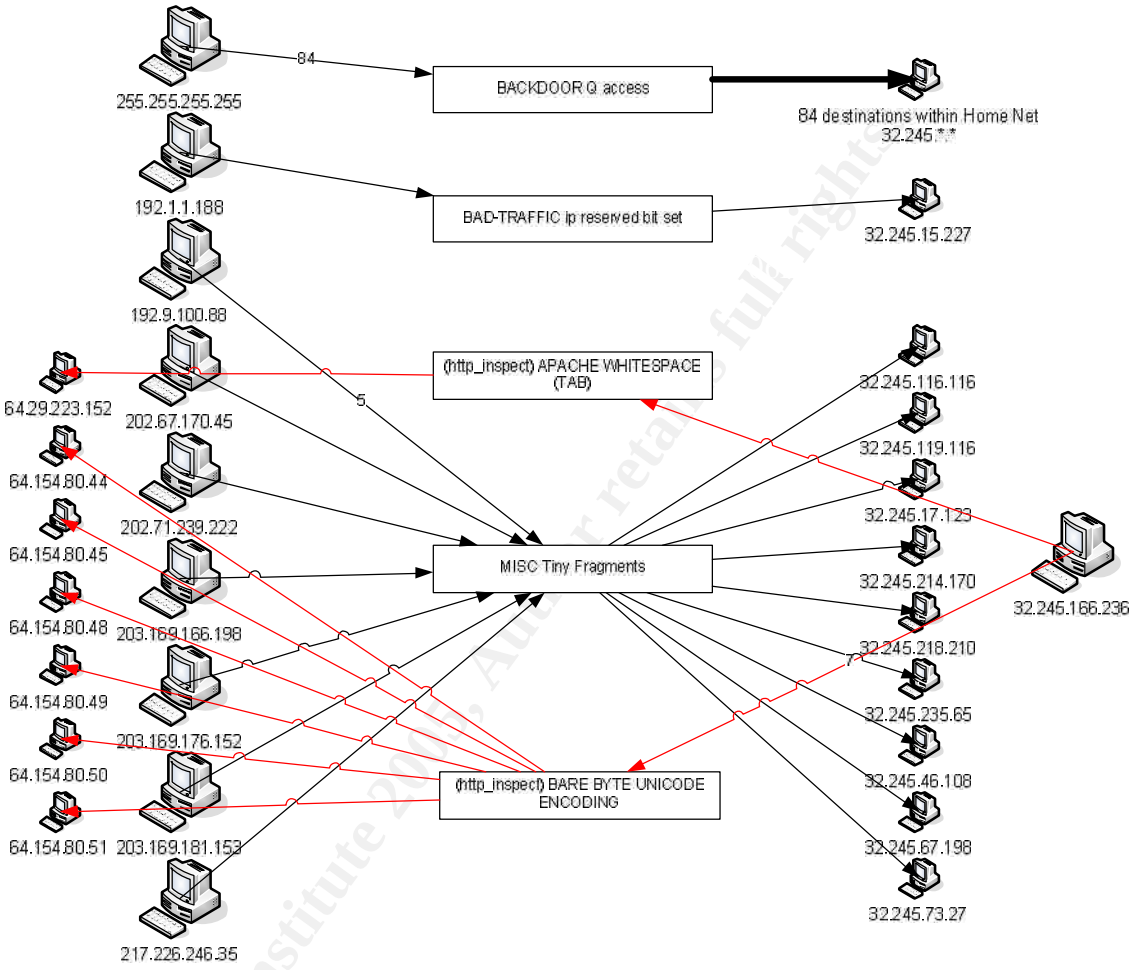


The above network topology is deduced from information gathered during the analysis of all of the traffic contained within the files *2002.9.19* and *2002.9.20*. It was immediately determined that the home, or protected, network was the IP range 32.245.0.0 because all traffic encountered in the files was either sourced from or destined to this range. The most prominent factor when establishing this diagram was the fact that all of the traffic within the files came from either one of two total MAC addresses. All external traffic inbound to the home network had an Ethernet source address of 00:03:E3:D9:26:C0, which, according to coffer[1], belongs to Cisco Systems. The internal MAC 00:00:0C:04:B2:33, also belonging to Cisco Systems as per coffer, was in turn the Ethernet source for all traffic outbound from the home network. Because the traffic only contained these two MAC addresses as the Ethernet source and destination it could be assumed that the IDS sensor resided between them. Following the placement of the IDS sensor it could be established that 32.245.166.236 was a proxy server because all outbound IP

[1] http://www.coffer.com/mac_find/

traffic had 32.245.166.236 as the source IP, except for a limited number of packets outbound from port 80 of 32.245.166.119, which turned out to be return code 403 (forbidden http), hence most likely a web server.

Diagram 2: Link Graph



This graph shows a visual representation of the alerts discussed in this paper. The black lines show traffic inbound to systems on the home network through the event generated by Snort®. Multiple packets from the originating host are noted by a number value attributed to the line of traffic. The red lines show traffic outbound from the Home Network in the same manner. This outbound traffic was added to give the viewer a sense of how the traffic flows in and out of the Home Network.

Overview of Detects

To generate the detects I merged the files *2002.9.19* and *2002.9.20*, both from http://isc.sans.org/logs/Raw/, using ethereal and named the merged files *2002.9.19.20*. 191 alerts were logged from a total of 20,300 packets. Below

is a pivot chart listing of all the signatures associated to the source and destination IPs of the packets that generated those signatures.

Sig msg	Source	Destination
(http_inspect) APACHE WHITESPACE (TAB)	32.245.166.236	64.29.223.152
(http_inspect) BARE BYTE UNICODE ENCODING	32.245.166.236	64.154.80.44
		64.154.80.45
		64.154.80.48
		64.154.80.49
		64.154.80.50
		64.154.80.51
		64.29.223.152
(http_inspect) NON-RFC HTTP DELIMITER	142.177.209.197	32.245.166.119
	203.204.175.123	32.245.166.119
	210.220.73.27	32.245.166.119
	212.98.224.2	32.245.166.119
	218.104.83.172	32.245.166.119
	218.18.14.52	32.245.166.119
BACKDOOR Q access	255.255.255.255	32.245.1.12
		32.245.11.174
		32.245.11.241
		78 dest. cut out to conserve space
		32.245.80.144
		32.245.80.228
		32.245.88.69
BAD-TRAFFIC ip reserved bit set	192.1.1.188	32.245.15.227
BAD-TRAFFIC tcp port 0 traffic	211.47.255.20	32.245.110.126
		32.245.208.55
MISC Tiny Fragments	192.9.100.88	32.245.17.123
		32.245.218.210
		32.245.235.65
		32.245.67.198
		32.245.73.27
	202.67.170.45	32.245.119.116
	202.71.239.222	32.245.214.170
	203.169.166.198	32.245.116.116
	203.169.176.152	32.245.119.116
	203.169.181.153	32.245.119.116
	217.226.246.35	32.245.46.108
SHELLCODE x86 setuid 0	207.188.7.147	32.245.166.236

Event counts can be fount in the chart below:

Sig msg	Count
(http_inspect) APACHE WHITESPACE (TAB)	1
(http_inspect) BARE BYTE UNICODE ENCODING	54
(http_inspect) NON-RFC HTTP DELIMITER	7
BACKDOOR Q access	84
BAD-TRAFFIC ip reserved bit set	1

BAD-TRAFFIC tcp port 0 traffic	32
MISC Tiny Fragments	11
SHELLCODE x86 setuid 0	1
total	**191**

General statistics:

Total alerts logged	191
Total unique alerts	8
Total unique sources	80
Total unique sources generating events	18
Total unique destinations	2133
Total unique destinations with event generated	105
Total unique internal destinations	2108
Total unique internal sources	2

Critical Detect 1 (MISC Tiny Fragments)

Description of the detect

Packet fragmentation is a well known function of TCP/IP communication discussed in detail in RFC 791[2]. The existence of fragmented packets has become a much more limited occurrence within modern networks because Ethernet's default MTU of 1500 is adequate for most applications. Traffic of this nature is always suspicious when discovered in a network. It may be an indication of a source trying to evade reconnaissance detection by splitting up the TCP header or evade detection of a malicious payload by sending fragmented TCP and UDP headers past the IDS or firewall. In support of this RFC 1858 states that "If the fragment size is made small enough to force some of a TCP packet's TCP header fields into the second fragment, filter rules that specify patterns for those fields will not match"[3] this is because many firewalls that see heavy traffic don't reassemble all fragments to check packet filtering. In addition to that RFC 1858 also states that "if the filtering router lies on one of several parallel paths, the filtering module will not see every fragment and cannot guarantee complete fragment filtering in the case of packets that should be dropped."[4]

Reason this detect was selected

This traffic is suspicious because if more data is to follow, indicated by the more fragment flag being set, then why wasn't that additional data packaged with this packet. Anytime the total size of a packet is 40 bytes or less I assume there is some kind of header manipulation taking place.

[2] http://rfc.net/rfc791.html
[3] http://rfc.net/rfc1858.html
[4] http://rfc.net/rfc1858.html

Detect was generated by

Snort 2.2 on Suse Linux 9.0 through VMWare partition in Windows XP with: Snort 2.2 rule set from Wed Dec 1st, 2004; all Rules enabled.

The command that generated these detects was:

```
snort -r c:\snort\raw\2002.9.19.20 -c c:\snort\rules\snort.conf -l c:\snort\log
```

The rule that generated these events from *misc.rules* file:

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC Tiny Fragments"; dsize:<
25; fragbits:M; classtype:bad-unknown; sid:522; rev:2;)
The specific variable that triggered these events is the use of source IP
255.255.255.255 as it falls within the 255.255.255.0 range.
```

The stimulus for this alert is the detection of an IP packet with the 'more fragments' flag set (`fragbits:M`) and a data size of less than 25 bytes (`dsize:< 25`).

The events generated:

```
[**] MISC Tiny Fragments [**]
10/18/02-19:02:07.776507 192.9.100.88 -> 32.245.235.65
TCP TTL:235 TOS:0x0 ID:0 IpLen:20 DgmLen:40 MF
Frag Offset: 0x1DB3   Frag Size: 0x0014
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 28 00 00 3D B3 EB 06 4B 6E C0 09 64 58 20 F5  .(..=...Kn..dX .
0x0020: EB 41 09 24 00 50 01 52 F5 C0 01 52 F5 C0 72 04  .A.$.P.R...R..r.
0x0030: 00 00 4F 98 00 00 00 00 00 00 00 00              ..O.........

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

[**] MISC Tiny Fragments [**]
10/19/02-01:51:58.976507 192.9.100.88 -> 32.245.67.198
TCP TTL:235 TOS:0x0 ID:0 IpLen:20 DgmLen:40 MF
Frag Offset: 0x1DB3   Frag Size: 0x0014
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 28 00 00 3D B3 EB 06 F5 E6 C0 09 64 58 20 F5  .(..=.......dX .
0x0020: 43 C6 13 84 00 50 02 CA 33 8C 02 CA 33 8C 04 04  C....P..3...3...
0x0030: 00 00 DF 2A 00 00 00 00 00 00 00 00              ...*........

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

[**] MISC Tiny Fragments [**]
10/19/02-15:27:48.046507 192.9.100.88 -> 32.245.218.210
TCP TTL:235 TOS:0x0 ID:0 IpLen:20 DgmLen:40 MF
Frag Offset: 0x1DB3   Frag Size: 0x0014
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 28 00 00 3D B3 EB 06 5C DB C0 09 64 58 20 F5  .(..=...\...dX .
0x0020: DA D2 07 C4 00 50 00 51 8B FA 00 51 8B FA 05 04  .....P.Q...Q....
0x0030: 00 00 A4 F4 00 00 00 00 00 00 00 00              ............

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

[**] MISC Tiny Fragments [**]
10/19/02-19:55:49.936507 192.9.100.88 -> 32.245.17.123
TCP TTL:235 TOS:0x0 ID:0 IpLen:20 DgmLen:40 MF
Frag Offset: 0x1DB3   Frag Size: 0x0014
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 28 00 00 3D B3 EB 06 28 32 C0 09 64 58 20 F5  .(..=...(2..dX .
0x0020: 11 7B 05 7C 00 50 01 46 EF 82 01 46 EF 82 05 04  .{.|.P.F...F....
0x0030: 00 00 A9 98 00 00 00 00 00 00 00 00              ............

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

[**] MISC Tiny Fragments [**]
10/20/02-23:29:37.106507 192.9.100.88 -> 32.245.73.27
```

```
TCP TTL:235 TOS:0x0 ID:0 IpLen:20 DgmLen:40 MF
Frag Offset: 0x1DB3   Frag Size: 0x0014
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 28 00 00 3D B3 EB 06 EF 93 C0 09 64 58 20 F5  .(..=.......dX .
0x0020: 49 1B 0A 97 00 50 02 0A AC 4C 02 0A AC 4C 70 04  I....P...L...Lp.
0x0030: 00 00 85 C3 00 00 00 00 00 00 00 00              ............

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

Probability the source address was spoofed

Performing a "who is" query from Arin[5] shows 192.9.100.88 as belonging to Sun Microsystems.

I went to dshield[6] to see to look up the source, but there were no records for that IP.

The prospect exists that the source was spoofed beginning with the possibility that the packets were crafted, but other than that there is no definitive proof. There are 5 total packets from 192.9.100.88, each of them generated the MISC tiny fragments alert. The size of each of these packets is 40 bytes, with 20 byte IP headers. So the remaining 20 bytes can be attributed to the TCP header.

The packets from 192.9.100.88 all had a high and uncommon TTL of 235, supporting the evidence that the packets were crafted. The only widely used OS with a default TTL value above 230 is Solaris 2.x[7] with a TTL of 255. The odds are that 255 was not the original TTL because 20 hops is quite some distance to go without having a packet of this nature discarded. However, since the TTL was the same for all packets over 2 day I have to assume that the source of this traffic is close to the perimeter of this network. Other than that I assume that the source is not being spoofed because the packets appear to have no valuable purpose for malicious intent.

Attack Mechanism

I ran the files through Windump to identify any additional information Ethereal missed. Ethereal does not show any TCP values because it stops calculating when it identifies Fragmented IP Protocol packets, as in this case.

Command:

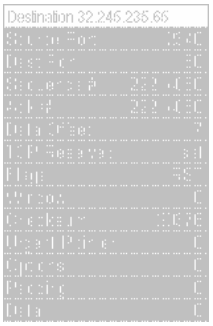
```
windump -e -S -X -vvv -r 2002.9.19.20 -n "src host 192.1.1.188"
```

Output:

```
19:02:07.776507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: 192.9.100.88 >
32.245.235.65: (frag 0:20@60824+) (ttl 2
```

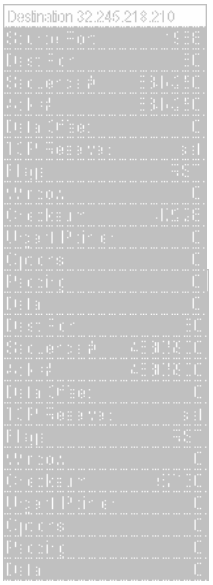
[5] http://www.arin.net/
[6] http://www.dshield.org/ipinfo.php
[7] http://www.giac.org/practical/GCIA/Ron_Shuck_GCIA.pdf

```
35, len 40, bad cksum 4b6e!)
0x0000   4500 0028 0000 3db3 eb06 4b6e c009 6458        E..(..=...Kn..dX
0x0010   20f5 eb41 0924 0050 0152 f5c0 0152 f5c0        ...A.$.P.R...R..
0x0020   7204 0000 4f98 0000 0000 0000 0000             r...O.........
01:51:58.976507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: 192.9.100.88 >
32.245.67.198: (frag 0:20@60824+) (ttl 2
35, len 40, bad cksum f5e6!)
0x0000   4500 0028 0000 3db3 eb06 f5e6 c009 6458        E..(..=.......dX
0x0010   20f5 43c6 1384 0050 02ca 338c 02ca 338c        ..C....P..3...3.
0x0020   0404 0000 df2a 0000 0000 0000 0000             .....*........
15:27:48.046507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: 192.9.100.88 >
32.245.218.210: (frag 0:20@60824+) (ttl
235, len 40, bad cksum 5cdb!)
0x0000   4500 0028 0000 3db3 eb06 5cdb c009 6458        E..(..=...\...dX
0x0010   20f5 dad2 07c4 0050 0051 8bfa 0051 8bfa        .......P.Q...Q..
0x0020   0504 0000 a4f4 0000 0000 0000 0000             ..............
19:55:49.936507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: 192.9.100.88 >
32.245.17.123: (frag 0:20@60824+) (ttl 2
35, len 40, bad cksum 2832!)
0x0000   4500 0028 0000 3db3 eb06 2832 c009 6458        E..(..=...(2..dX
0x0010   20f5 117b 057c 0050 0146 ef82 0146 ef82        ...{.|.P.F...F..
0x0020   0504 0000 a998 0000 0000 0000 0000             ..............
23:29:37.106507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: 192.9.100.88 >
32.245.73.27: (frag 0:20@60824+) (ttl 23
5, len 40, bad cksum ef93!)
0x0000   4500 0028 0000 3db3 eb06 ef93 c009 6458        E..(..=.......dX
0x0010   20f5 491b 0a97 0050 020a ac4c 020a ac4c        ..I....P...L...L
0x0020   7004 0000 85c3 0000 0000 0000 0000             p.............
```



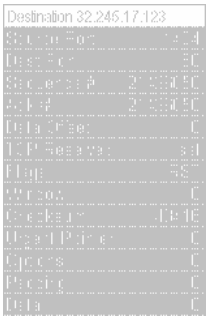
I overlaid the Hex dump onto a TCP header and you can see the TCP values along the right.

You will notice in the TCP header settings to the right that there are few consistencies, but many irregularities. The source ports/sequence #s/IHLs and checksums all change from packet to packet. The IHL, or data offset, should be a minimum value of 5 for a complete TCP header, but the fragmentation does not take place before this field. Here we have 2 packets with a valid value of 7 and the other 3 with an invalid value of 0. The TCP header info is oftentimes erroneous in these packets and TCP reserved bits are set intermittently.



The only real consistency between the packets is the use of destination port 80 (HTTP) and the setting of the RST flag. I checked and found that none of the destination IPs resolve to a web page. I read a thread where the user was encountering a lot of TCP data Offsets set less than 5[8]. In that case a majority of that traffic was being cause by traffic from a modem pool. I am wondering if in this case this traffic might also be from a similar pool.

The MISC tiny fragments are meant to be an evasive attack, but in this case the attacker has failed miserably as



[8] http://www.mcabee.org/lists/snort-users/Dec-03/msg00632.html

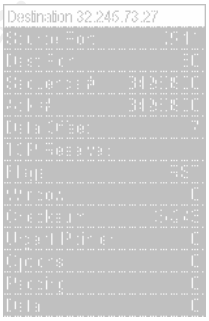
there are many flags that raise reason for question in regards to these packets. I believe these packets to be corrupt from a dial-up pool of some kind.

Correlations

GCIA Practical Assignments
http://www.giac.org/practical/GCIA/Doug_Kite_GCIA.pdf

Snort Rule Documentation
http://www.snort.org/snort-db/sid.html?sid=522

RFCs
http://rfc.net/rfc791.html
http://rfc.net/rfc1858.html



TCP Data offset less than 5
http://www.mcabee.org/lists/snort-users/Dec-03/msg00632.html

Fragmentation Attack analysis
http://www.inet-sec.org/docs/DoS/fragma.html

Dshield intrusions (MISC tiny fragments)
http://www.dshield.org/pipermail/intrusions/2003-April/007375.php

Evidence of active targeting

Besides the fact that 5 packets from the same source are all destined to IPs in the Home network I don't feel there is any active targeting taking place here. The events were generated in no chronological or sequential order, but that could be done to avoid detection, which would be believable if not for all of the irregularities associated with these packets.

Severity

(3 + 1) - (3 + 3) = 4 – 6 = -2

Criticality

3

Traffic was directed to 5 different hosts in the network, but I have no specific information regarding the role of any of those systems.

Lethality

1

If the packets were maliciously crafted then there are a number of malicious tasks that may be performed. However, I believe this traffic to be corrupt and therefore benign to the destination hosts.

System countermeasures

3

It is unknown what processes are running on the destination hosts, nor what processes are supposed to be running. There could also be virus scanning software and personal firewalls on the destination hosts that could assist in this case. However, without additional network information an accurate gauge of system countermeasures cannot be obtained.

Network countermeasures

3

The network I inferred has a firewall, IDS, and proxy server; which is a good start to a secure stance. However, being that we have no information regarding firewall rules or specific system functions a truly accurate gauge of criticality cannot be obtained.

Defensive Recommendations

Enforce rules that require the minimum size of the first fragment to be great enough to contain all of the protocol header information. Also, block external traffic with the TCP reserved bits set at the firewall.

Critical Detect 2 (BACKDOOR Q access)

Description of the detect

The 'BACKDOOR Q access' signature is generated when a TCP packet is observed by the IDS with the source IP of 255.255.255.0 The signature's stimulus is somewhat general, so there exists the possibility that Snort might be incorrectly classifying this traffic. The generation of the signature denotes that a packet has been discovered which was generated by the Q remote administration tool. The purpose of this packet would be as a probe of sorts sent out to activate any "server" installations of Q. This tool, originally written by "Mixter" (http://mixter.void.ru/), may allow an attacker to control a remote system, or "server", from a "client" installation on the attacker's machine. Q is oftentimes referred to as a "Trojan Horse" but only truly assumes that designation when it is unknowingly installed on the target system as a means for the attacker to gain unauthorized access, and/or compromise the security of that system in any way. Otherwise Q can be looked at as "part of a hacker's root-kit and deployed as a fairly secure back-door means of accessing an already compromised system"[9].

[9] http://www.sans.org/resources/idfaq/qtrojan.php

Reason this detect was selected

This detect was selected because of definitive proof that the packet was crafted and may be being used in conjunction with a backdoor Trojan. The security ramifications associated with a possible infection within the local network are substantial.

Detect was generated by

Snort 2.2 on Suse Linux 9.0 through VMWare partition in Windows XP with:
Snort 2.2 rule set from Wed Dec 1st, 2004; all Rules enabled.

The command that generated these detects was:

```
snort -r c:\snort\raw\2002.9.19.20 -c c:\snort\rules\snort.conf -l c:\snort\log
```

The rule that generated these events from *backdoor.rules* file:

```
alert tcp 255.255.255.0/24 any -> $HOME_NET any (msg:"BACKDOOR Q access";
flow:stateless; dsize:>1; flags:A+; reference:arachnids,203; classtype:misc-
activity; sid:184; rev:7;)
```

The stimulus for this alert is the detection of a TCP packet with source IP 255.255.255.255, as that IP falls within the 255.255.255.0 range (`alert tcp 255.255.255.0`).

The events generated:

```
[**] BACKDOOR Q access [**]
10/18/02-17:35:32.046507 255.255.255.255:31337 -> 32.245.80.13:515
TCP TTL:15 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 0F 06 26 B5 FF FF FF FF 20 F5  .+......&..... .
0x0020: 50 0D 7A 69 02 03 00 00 00 00 00 00 00 00 50 14  P.zi..........P.
0x0030: 00 00 DB DC 00 00 63 6B 6F 00 00 00              ......cko...

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

[**] BACKDOOR Q access [**]
10/18/02-17:36:19.996507 255.255.255.255:31337 -> 32.245.177.213:515
TCP TTL:15 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 0F 06 C3 EB FF FF FF FF 20 F5  .+............ .
0x0020: B1 D5 7A 69 02 03 00 00 00 00 00 00 00 00 50 14  ..zi..........P.
0x0030: 00 00 79 13 00 00 63 6B 6F 00 00 00              ..y...cko...

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

[**] BACKDOOR Q access [**]
10/18/02-18:07:26.076507 255.255.255.255:31337 -> 32.245.80.144:515
TCP TTL:15 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 0F 06 27 30 FF FF FF FF 20 F5  .+......'0.... .
0x0020: 50 90 7A 69 02 03 00 00 00 00 00 00 00 00 50 14  P.zi..........P.
0x0030: 00 00 DC 57 00 00 63 6B 6F 00 00 00              ...W..cko...

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

[**] BACKDOOR Q access [**]
10/18/02-18:07:53.036507 255.255.255.255:31337 -> 32.245.232.125:515
TCP TTL:15 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 0F 06 8D 43 FF FF FF FF 20 F5   .+.......C.... .
0x0020: E8 7D 7A 69 02 03 00 00 00 00 00 00 00 00 50 14   .}zi..........P.
0x0030: 00 00 42 6B 00 00 63 6B 6F 00 00 00               ..Bk..cko...

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

Probability the source address was spoofed

Performing a "who is" query from Arin[10] provides no additional information and shows 255.255.255.255 as belonging to IANA. This is expected as this IP is an illegitimate source IP and therefore owned by no one.

I went to dshield[11] to see if there was any additional info there, but there were no records for that IP.

The source IP 255.255.255.255 is not a valid IP address. It could never be made to generate traffic nor accept returned traffic from any destination not within the local network. For these reasons the source is definitely being spoofed and the packet has been crafted.

Attack Mechanism

According to RFC 917 "A destination address of all ones (255.255.255.255) causes a datagram to be sent as a broadcast on the local physical network; it must not be forwarded by any gateway."[12] So the IP 255.255.255.255 as a destination is non-routable, however, what RFC 917 does not explicitly discuss is the use of 255.255.255.255 as the source IP. RFC 919 makes a more general statement with "The address 255.255.255.255 denotes a broadcast on a local hardware network, which must not be forwarded. This address may be used, for example, by hosts that do not know their network number and are asking some server for it,"[13] but then goes on to say that a host on the network may "broadcast to all of its immediate neighbors by using 255.255.255.255," obviously suggesting that it only refers to the use of 255.255.255.255 as a destination address.

The use of TCP flags RST/ACK is also unusual. A packet with the RST/ACK flag denotes that it is sent in reply to a SYN packet that was directed to a closed port. Therefore a SYN packet would have to be sent from the home network first for 255.255.255.255 to elicit this type of reply. Since the destination IP 255.255.255.255 is not valid, the natural occurrence of this packet is impossible.

Generally a TTL value can be useful for a couple of reasons. The obvious reason is that it can tell you the distance between the packet logging device

[10] http://www.arin.net/
[11] http://www.dshield.org/ipinfo.php
[12] http://rfc.net/rfc917.html
[13] http://rfc.net/rfc919.html

and the source host. If you have pre-calculated that distance by measuring the hops to the source you can use the sum of that number plus the packet's TTL to get the default TTL value. That sum is the original TTL of the system that sent the packet initially. By cross-referencing that sum against a list of Default TTL values per OS you may be able to fingerprint the OS of the source system. The TTL value is 15 for every event in this case. Most modern operating systems use a default TTL value of 32, 64 or 128[14]. So, the value of 15 is not much help in this case. Being that there were no IP options setting strict source routing, the fact that the value was unfailingly 15 over 2 days and 84 packets indicates that either the path it takes from the source host is very concrete or the source host resides close to the border of this network.

The TCP sequence #'s of each packet is 0 and this is not a relative sequence number since no three-way handshake has taken place. Being that, "The TCP specification in RFC 793 suggests that the generation of the ISN be bound to a 32-bit clock that increments once every 4 microseconds (that is 250,000 per second)"[15] the chances that each one of these packets would legitimately generate a sequence number of 0 over a 2 day period is Statistically impossible.

The source port of 31337 draws attention. This port is better known as the eleet port (notorious for its use by Cult of the Dead Cow with the BackOrifice Trojan). Any use of this port should garner additional interest as it is a favorite port of "hackers". The destination port is 515 (the line printer or lpr port: the primary port for UNIX printing services), I can think of no instances in which port 515 need be open to systems outside the local network.

The payload of "cko" could be a stimulus to activate the Q "server" program, which would at that point return instruction to a "client" host designated in the Q conf file, but for that to work the destination would need to have the Q process already up and running. Therefore, the destination hosts could have the Q "server" installed, however, without further information I cannot say so conclusively.

These events are directed to hosts all over the home network in no discernable chronological or sequential order. I tend to believe that this traffic is an unsuccessful modification of Q intended to work as a Trojan.

Correlations

Mixter's webpage (to obtain code and download Q):
http://mixter.void.ru/

GCIA Papers:

[14] http://secfr.nerim.net/docs/fingerprint/en/ttl_default.html
[15] http://www.camtp.uni-mb.si/books/Internet-Book/TCP_ISN.html

http://www.giac.org/practical/GCIA/Al_Maslowski-Yerges_GCIA.pdf
http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc
http://www.giac.org/practical/GCIA/Tillman_Hodgson_GCIA.pdf

Snort signature database:
http://www.snort.org/snort-db/sid.html?sid=1-184

Security vendors:
http://www.f-secure.fi/v-descs/backdoor.shtml

Default TTL Values:
http://secfr.nerim.net/docs/fingerprint/en/ttl_default.html

Mailing Lists:
http://www.securityfocus.com/archive/75/182244/2003-05-11/2003-05-17/1
http://www.securityfocus.com/archive/75/182244/2001-04-28/2001-05-04/0

Sans FAQ:
http://www.sans.org/resources/idfaq/qtrojan.php

Sequence number selection:
http://www.camtp.uni-mb.si/books/Internet-Book/TCP_ISN.html

Evidence of active targeting

There is no evidence to support the theory that active targeting is taking place as the events were generated in no chronological or sequential order. The source may be targeting the home network being that packets were directed to 84 different hosts within the network, but at no host more than one time. The fact that that events came in so erratically could have been done to avoid detection, however, to reiterate, there is no specific evidence to support this.

Severity

(3 + 5) - (3 + 3) = 8 – 6 = 2

Criticality

3

Traffic was directed to 84 different hosts in the network, but I have no specific information regarding the role of any of those systems.

Lethality

5

If Q is installed on a system in the home network and it is working correctly, meaning that an external “client” host can activate the internal “server”

installation, then that system is 100% compromised and any number of malicious tasks may be conducted on and from that host.

System countermeasures

3

It is unknown what processes are running on the destination hosts, nor what processes are supposed to be running. Therefore, without additional network information an accurate gauge of system countermeasures cannot be obtained.

Network countermeasures

3

The network I inferred has a firewall, IDS, and proxy server; which is a good start to a secure stance. However, being that we have no information regarding firewall rules or specific system functions a truly accurate gauge of criticality cannot be obtained.

Specific Defensive Recommendations for Backdoor Q

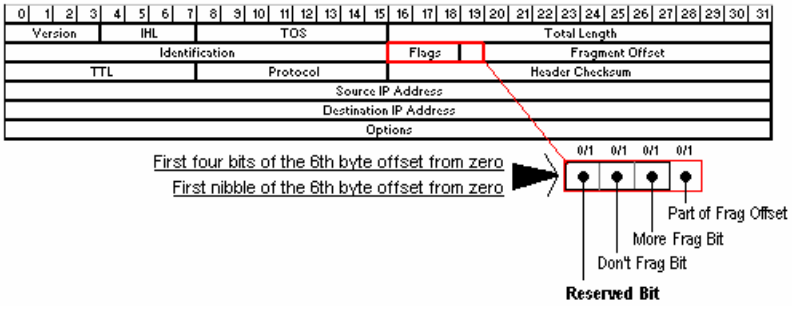
External traffic from IP 255.255.255.255 is never legitimate traffic, ensure that the 255.0.0.0 range is blocked by the firewall. Printers rarely, if ever, need to be accessed from the internet; therefore it is also recommended that inbound Port 515 traffic be shunned.

Critical Detect 3 (BAD-TRAFFIC ip reserved bit set)

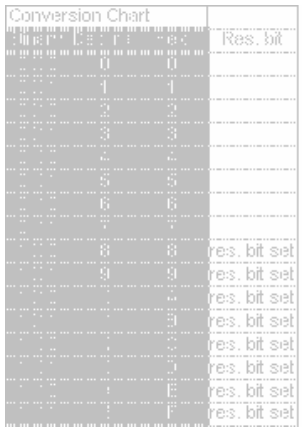
Description of the detect

The IP reserved bit was first brought to my attention in reference to RFC 3514[16]. This RFC was intended as an April fools prank by Steven M. Bellovin of AT&T. The resulting misinterpretations and responses were quite humorous.

Figure 1



The reserved bit is located at the 6th byte offset from 0 of the IP header (see figure 1). If that byte's first hexadecimal nibble value is 8 (0x8) or greater then the reserved bit is set. If you look at the hex conversion chart to the right you will see the breakdown for each nibble or hexadecimal value. 1 byte = 2 hex nibbles or 8 binary bits,



[16] http://rfc.net/rfc3514.html

therefore 1 nibble = 4 binary bits. Each set of 4 binary bits has 16 possible decimal values, from 0 to 15, in hex 0 to F. This bit is not used under normal conditions, and any use of the reserved bit means that the packet has been crafted.

Reason this detect was selected

The reserved bit being set is either proof of packet crafting or faulty NIC configuration; and there are no false positives associated with traffic of this nature. Different OS fingerprinting and reconnaissance methods have been associated with the use of the reserved bit, as well the suggestion that it may be used as a stimulus to "instigate covert channel communications"[17]. I selected this detect because of the sure-fire evidence of packet manipulation.

Detect was generated by

Snort 2.2 on Suse Linux 9.0 through VMWare partition in Windows XP with: Snort 2.2 rule set from Wed Dec 1st, 2004; all Rules enabled.

The command that generated these detects was:

```
snort -r c:\snort\raw\2002.9.19.20 -c c:\snort\rules\snort.conf -l c:\snort\log
```

The rule that generated these events from the *bad-traffic.rules* files:

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"BAD-TRAFFIC ip reserved bit
set"; fragbits:R; classtype:misc-activity; sid:523; rev:5;)
```

The stimulus for this alert is found in fragbits field with a value of R (`fragbits:R`). This means that the reserved bit is set.

The event generated:

```
[**] BAD-TRAFFIC ip reserved bit set [**]
10/19/02-03:49:10.256507 192.1.1.188 -> 32.245.15.227
TCP TTL:230 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset: 0x0864   Frag Size: 0x0014
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 28 00 00 88 64 E6 06 46 BD C0 01 01 BC 20 F5  .(...d..F..... .
0x0020: 0F E3 05 AA 00 50 4D 9B 8B 22 4D 9B 8B 22 00 04  .....PM.."M.."..
0x0030: 00 00 42 BD 00 00 00 00 00 00 00 00              ..B.........

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

Probability the source address was spoofed

Performing a "who is" query on Arin[18] shows 192.1.1.188 as belonging to BBN or Bolt Beranek and Newman Inc.

[17] http://www.dshield.org/pipermail/intrusions/2003-May/007635.php
[18] http://www.arin.net/

I went to dshield[19] to see if there was any additional info there, but there were no records for that IP.

The prospect exists that the source was spoofed beginning with the possibility that the packet was crafted, but other than that there is no definitive proof.

If the purpose of this traffic is to perform reconnaissance then there is a need for the source not to be spoofed so that the source host can receive return traffic from the destination host (being that no source routing options were enables in this packet). If, on the other hand, this traffic is meant to be a stimulus for the initiation of covert communication then the need to not spoof is not so definite because an application on the destination host could prompt the communication after this special packet triggers it to launch. However, in a case such as that a process would need to be running on the destination host that has instructions for handling packets of this nature.

I did some additional research online by searching from Google and yahoo for 192.1.1.* and found some interesting additional information about this address range. RFC 1166[20] lists that range as belonging to BBN, as does Arin, but specifically lists that the 192.1.1.* range is used for BBN Local Nets. This rfc, on the other hand, is from 1990, so I can't put too much weight in it. I also noticed from my web search that many people use the 192.1.1.* range "to form private corporate networks" and "Use of the 192.1.1 Class C network for this purpose is an Internet folk tradition"[21]. This suggests that this packet might have come from such a network, but there is no way to tell this for certain without a full network trace of all traffic. A full network trace would be useful in determining if spoofing is taking place. This would allow me to identify any return traffic from the destination host at around the same time as this original packet. But the use of a proxy and the lack of such a network trace prevent me from making these observations; as a result I believe that the source was not spoofed if only because the communication was limited to a single packet.

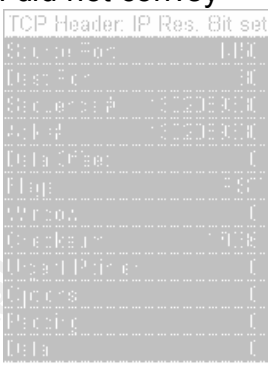
The following analysis was provided before I realized that Ethereal was not reading the TCP header:

> The use of source port 0 gives some indication of whether or not the source host wants to receive return traffic, which could help when determining whether spoofing is taking place. This is because if the destination is OpenBSD then the packet will be discarded, whereas Windows and Linux will reply to such traffic[22]. However, without knowledge of the destination OS this information is not useful. A full list of

[19] http://www.dshield.org/ipinfo.php
[20] http://rfc.net/rfc1166.html
[21] http://www.byte.com/art/9511/sec8/art2.htm
[22] http://www.networkpenetration.com/port0.html

port 0 fingerprints can be obtained from URL http://www.networkpenetration.com/port0.html.

After some additional investigation I realized that Ethereal did not convey the traffic correctly. Ethereal reads these packets as fragmented IP protocol packets and does no further analysis. When I noticed this I overlaid the hex-dump onto a TCP header and did the math. I will leave my port 0 analysis information in my detect analysis because it is both useful information when analyzing traffic and an excellent reminder that all the tools in the world don't necessarily read the information as it is meant to be read. To the right you can read the TCP header settings.



Attack Mechanism

The packet had a high and uncommon TTL of 230, supporting the evidence that the packet was crafted. The only widely used OS with a default TTL value above 230 is Solaris 2.x[23] with a TTL of 255. The odds are that 255 was not the original TTL because 25 hops is quite some distance to go without having a packet of this nature discarded.

The source port of 1450 is a low ephemeral port oftentimes associated to "Tandem Distributed Workbench Facility". All of my research on this product turned up no evidence that it uses the IP reserved bit for any of its functions.

The RST flag was set. This indicates that the source host was not seeking any kind of return traffic from the destination, which most likely rules out standard reconnaissance as the reason for packet generation. This fact points toward either the use of the reserved bit as a stimulus mechanism for a covert program of some nature, or merely as a corrupt packet.

To see if there was any additional info in the packet that Ethereal might have missed I ran the packet through windump with the following command and output:

Protocol ID:

```
IP
TCP
```

Command:

```
windump -e -S -X -vvv -r 2002.9.19.20 -n "src host 192.1.1.188"
```

Output:

```
03:49:10.256507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: 192.1.1.188 >
32.245.15.227: (frag 0:20@17184) (ttl 230, len 40, bad cksum 46bd!)
0x0000   4500 0028 0000 8864 e606 46bd c001 01bc        E..(...d..F.....
0x0010   20f5 0fe3 05aa 0050 4d9b 8b22 4d9b 8b22        .......PM.."M.."
```

[23] http://www.giac.org/practical/GCIA/Ron_Shuck_GCIA.pdf

```
0x0020   0004 0000 42bd 0000 0000 0000 0000          ....B.........
```

As in Ethereal the checksum is known to be bad because of the scrubbing that took place on these files before they were posted by SANS.

The fact that the TCP Acknowledgement number is the same as the TCP Sequence number indicates nothing. Windump reads that this packet contains data bytes 0 to 20 of a 17184 byte payload, which is the TCP header in this case, but the more fragments flag is not set. The payload is derived from a decimal representation of the IP fragment offset multiplied by 8 because it is represented in units of 8 bytes. This information also tells me very little other than the fact that this packet is indicating that more traffic should be coming, but there are no remaining fragments. The TCP offset, or TCP Header Length, is an invalid # at 0, a legitimate TCP header should have a value of no less than 5 (32 bit words). All of this evidence supports the case that the packet was either crafted or has been corrupted.

Nmap and Hping2 are capable of crafting packets with the reserved bit set, yet the logic for crafting in those cases is as reconnaissance, requiring return traffic to make any conclusions about the destination host.

Correlations

GCIA Papers:
http://www.giac.org/practical/GCIA/Ryan_Barrett_GCIA.pdf
http://www.giac.org/practical/GCIA/Brian_Granier_GCIA.pdf

Snort signature database:
http://www.snort.org/snort-db/sid.html?id=523

Dshield Intrusions
http://www.dshield.org/pipermail/intrusions/2003-May/007635.php

NMAP documentation
http://www.insecure.org/nmap/data/nmap_manpage.html

TCP RST flag usage
http://rfc.net/rfc3360.html

Evidence of active targeting

There is no specific evidence of active targeting in this case other than the presence of a single packet directed to a sole destination. That being the case one would have to assume that the destination host was targeted in this case, the dilemma in this case is whether the packet is malicious or merely corrupt.

Severity

(2 + 5) - (3 + 3) = 7 – 6 = 1

Criticality

2

Traffic was directed to a single host within the network, but I have no specific information regarding the role of that host. Without additional network information an accurate gauge of criticality cannot be obtained.

Lethality

5

The "BAD-TRAFFIC ip reserved bit set" indicates one of two things in this case; Corrupt packet or application stimulus, because the reconnaissance uses were ruled out due to the use of the RST flag. The destination application that I speak of could be anything from a remote admin tool to a backdoor communication pipeline. Applications of this nature would render a system fully compromised and there would be no limits to the lethality in such a case.

System countermeasures

3

It is unknown what processes are running on the destination hosts, nor what processes are supposed to be running. Therefore, without additional network information an accurate gauge of system countermeasures cannot be obtained.

Network countermeasures

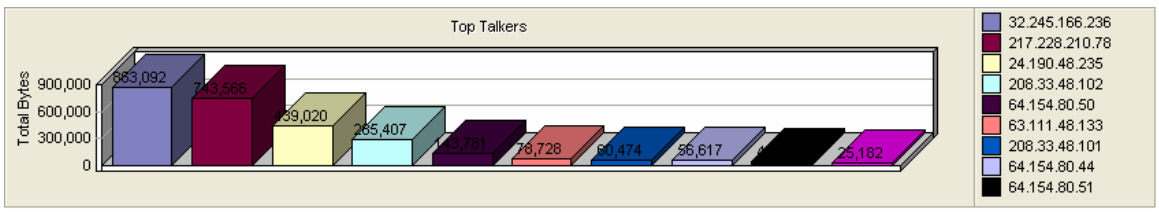
3

The network I inferred has a firewall, IDS, and proxy server; which is a good start to a secure stance. However, being that we have no information regarding firewall rules or specific system functions a truly accurate gauge of criticality cannot be obtained.

Specific defensive recommendations for IP reserved bit set

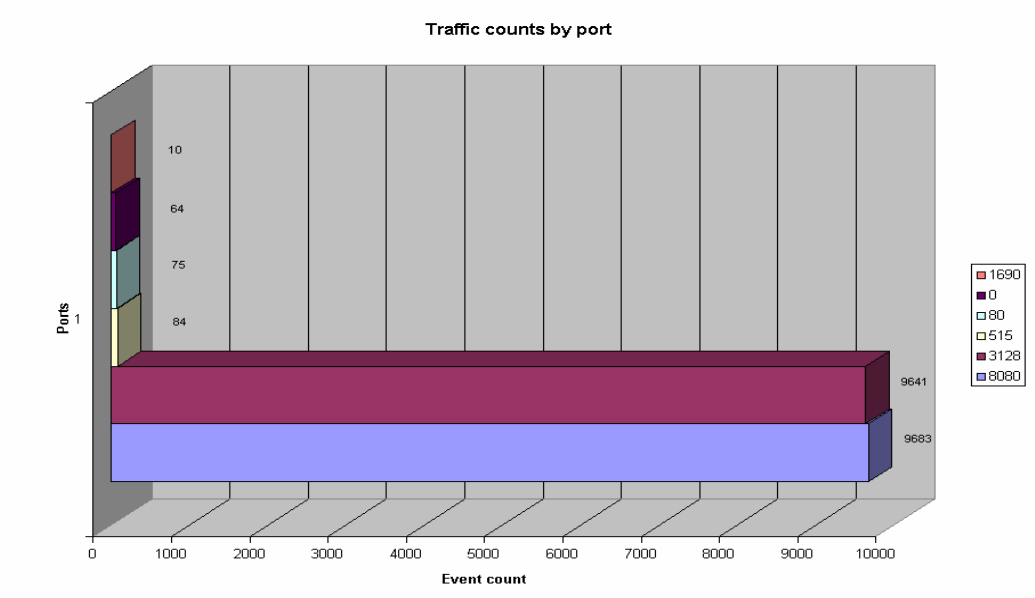
Block all external traffic with the reserved bit set at the firewall.
Ensure virus definitions are up to date on the destination system.
Ensure a personal firewall application is installed and running on the destination host.

Top Talkers

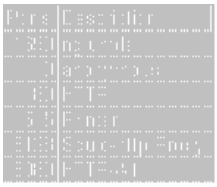


This chart was created using the trial version of Distinct Network Monitor. It reads the total bytes sent per IP, limiting the output to the top 10 IPs. The color legend is on the right.

Top Destination Ports



These ports were the top 6 ports targeted by external addresses. The port 0 traffic was caused by fragmented TCP headers. I created this graph using Microsoft Excel©. The ports descriptions are in the chart to the right.



3 Most Suspicious External source IPs

A majority of the traffic generated can be attributed to scans from external IPs 217.228.210.78 and 24.190.48.235. Those 2 hosts and one other, 148.63.228.0, are IPs I would like to research further as I believe their activity most suspicious.

1) 148.63.228.0 – ARIN WHOIS lookup: Starband Communications, Inc. – A series of TCP retransmissions from a network ID, 148.63.228.0, the context from the data of all of these packets reads “GIVE 186212675”, possible password transmission in clear text. The IP itself is invalid as IPs with 0 for the last octet are reserved for the network address.

2) 217.228.210.78 – RIPE WHOIS lookup: Deutsche Telekom - performed a fast and large SYN scan on Home Network. Deutche Telekom is Europe’s largest telecommunications company and a hotbed for malicious activity and reconnaissance.

3) 24.190.48.235 – ARIN WHOIS lookup: Optimum Online Cablevision Systems) - performed a fast and large SYN scan on Home Network. A script kiddie or compromised host is the likely cause of this very noisy traffic.

Part III – Analysis Process

Overview of Detects Generated

Workstation: IBM Thinkpad A22m with 800Mhz processor and 256 meg of RAM.
Environment: Suse 9.0 Professional running on Vmware partition within Windows XP installation.

1.) I went to the Suse ftp site[24] and downloaded the Snort 2.2 RPM then installed Snort using Yast.

2.) I downloaded the files *2002.9.19* and *2002.9.20* from the SANS raw download site[25].

3.) Using Ethereal 0.10.7© in Windows XP© I opened the file *2002.9.20*. After the file was loaded I then used Merge option under the Ethereal file menu and selected the file *2002.9.20*. (The merge is done in this order (from most to least recent) to prevent any issues with the time representation in Ethereal.)

4.) I saved the Merged Ethereal file as *2002.9.19.20* onto my hard drive. I attached this file to my web mail and mailed it to myself.

5.) I downloaded *2002.9.19.20* from my web mail in my Suse partition.

6.) I edited the snort.conf file and:
a) enabled all rules
b) added line `config checksum_mode: none` so events would be generated
c) added line `output alert_csv: snort.csv default` so all output would be in CSV format

7.) Ran command: `snort -r c:\snort\raw\2002.9.19.20 -c c:\snort\rules\snort.conf -l c:\snort\log`
Snort switches:
-r read from file
-c specify a snorf.conf file
-l log to directory

8.) Edited the snort.conf file again and:

[24] http://ftp.lug.ro/suse/people/kssingvo/unsupported/snort/
[25] http://isc.sans.org/logs/Raw/

a) edited line `output alert_csv: snort.csv` to read `output alert_csv: ip.csv`
b) added rule `alert ip any any <> any any (msg:"ip traffic";)`

9.) Re-ran command: `snort -r c:\snort\raw\2002.9.19.20 -c c:\snort\rules\snort.conf -l c:\snort\log`

10.) Opened snort.csv and IP.csv in Microsoft Excel® and inserted the below values as the first line, saved both files. (line below is comma delimited)

```
timestamp, sig_generator, sig_id, sig_rev, msg, proto, src, srcport, dst, dstport,
ethsrc, ethdst, ethlen, tcpflags, tcpseq, tcpack, tcplen, tcpwindow, ttl, tos, id,
dgmlen, iplen, icmptype, icmpcode, icmpid, icmpseq
```

11.) Opened snort.csv in BrioQuery Explorer®, created a pivot chart and made `msg, src, srcport, dst, dstport, timestamp, ethsrc, ethdst` the labels. Sorted these values in different orders to ascertain patterns and perform analysis.

12.) Noticing that there were only 2 `ethsrc` and `ethdst` values (MAC addresses) I began to make my topology diagram of the network using Microsoft Visio® noting that the IDS sensor existed between 2 devices in the network, one separating it from the external domain, and one separating it from the internal network. The prevalence of all outbound traffic from one source indicated that that IP most likely belonged to the Proxy server.

13.) I generated my top 10 talkers graph using the program: Distinct Network Analyzer©.

14.) I generated my top 5 ports graph using Microsoft Excel©.

Endnotes References:

Gordon, Les "What is the Q trojan?" SANS Intrusion Detection FAQ. URL: http://www.sans.org/resources/idfaq/qtrojan.php (12 Nov. 2004)

Mogul, Jeffrey "INTERNET SUBNETS" Network Working Group. Oct 1984. URL: http://rfc.net/rfc917.html (12 Nov. 2004)

Mogul, Jeffrey "BROADCASTING INTERNET DATAGRAMS" Network Working Group. Oct 1984. URL: http://rfc.net/rfc919.html (12 Nov. 2004)

Northcutt, Stephen. Zeltser, Lenny. Winters, Scott. Frederick, Karen Kent. Ritchey, Ronald W. Inside Network Perimeter Security: The Definitive Guide to Firewalls, VPNs, Routers, and Intrusion Detection Systems. Jan 2001.

Dhanjani, Nitesh. HACK NOTES: Linux and Unix Security. 2003.

O'Dea, Michael. HACK NOTES: Windows Security. 2003

Stevens, Richard W. TCP/IP Illustrated Volume 1. 1994.

Northcutt, Stephen. Novak, Judy. Network Intrusion Detection Third Edition September 2002.

Initial Sequence Number Selection. URL: http://www.camtp.uni-mb.si/books/Internet-Book/TCP_ISN.html

Default TTL Values in TCP/IP. URL: http://secfr.nerim.net/docs/fingerprint/en/ttl_default.html

Bellovin, S. "The Security Flag in the IPv4 Header" Network Working Group. 1 Apr 2003. URL: http://rfc.net/rfc3514.html