



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Network Monitoring and Threat Detection In-Depth (Security 503)"
at <http://www.giac.org/registration/gcia>

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment
Version 4.0

Adam Kliarsky

December, 2004

© SANS Institute 2005, Author retains full rights.

Part I – Executive Summary	3
Purpose	3
Scope	3
Methods	3
Results	3
Conclusions	3
Recommendations	3
Part II – Detailed Analysis	4
Summary of Alerts:	12
Detect 1: BACKDOOR Q access	13
Description of detect:	13
Reason this detect was selected:	13
Detect generated by:	13
Probability the source address was spoofed:	15
Attack mechanism:	15
Correlations:	16
Evidence of active targeting:	16
Severity	17
Detect 2: WEBROOT DIRECTORY TRAVERSAL	18
Description of detect:	18
Reason this detect was selected:	18
Detect generated by:	18
Probability the source address was spoofed:	19
Attack mechanism:	19
Correlations:	23
Evidence of active targeting:	23
Severity	23
Detect 3: SHELLCODE x86 0xEB0C NOOP	24
Description of detect:	24
Reason this detect was selected:	24
Detect generated by:	24
Probability the source address was spoofed:	25
Attack mechanism:	25
Correlations:	28
Evidence of active targeting:	28
Severity	29
Network Statistics	30
Correlations	33
Insights into internal machines	34
Defensive recommendations	34
Part III – Analysis Process	35
Works Cited	36

Part I – Executive Summary

An investigative analysis was recently performed on the network of a University campus. The analysis was conducted in a postmortem manner, using binary capture files.

Purpose

This analysis was conducted to investigate and assess the types of network traffic anomalies seen on the University's local area network. An in-depth analysis is vital to ensure the integrity of the internal systems, University resources, and student records. Furthermore, a proactive approach to security is essential due to new legislative requirements (SB1386 for example), requiring the public disclosure of a compromise within an organization, a proactive approach to security is essential.

Scope

The analysis was performed on binary network captures taken from an intrusion detection system logging anomalous traffic between the ISP and the LAN. The dates of the traffic captured span 11/15/2002 through 11/18/2002. It is important to note that the scope of the analysis was limited to the capture files only, as no other information was presented regarding the architecture and technology deployed within the environment.

Methods

A combination of analytic methods was used, including both technical and non-technical programs and utilities. These utilities range from network analyzers to office productivity programs designed to structure and present the technical data in a clear, concise format.

Results

There were 718 alerts (16 unique) generated during the three days, representing a combination of malicious and benign traffic. High amounts of ingress traffic passed the border router destined for the campus LAN, however limited egress traffic was seen in response.

Conclusions

The University has several points of exposure on the network and needs to be diligent in safeguarding its systems and information. The University's technical staff needs to implement strict measures on network systems to mitigate the threats, including ingress/egress filtering on perimeter devices and system hardening.

Recommendations

The technical staff needs to closely monitor the alerts and tune the IDS where possible, as the time spent on false alarms can be overwhelming, and can obfuscate potential true attacks. Extra attention should be given to the targets of the attacks as well, to ensure systems are properly configured.

Part II – Detailed Analysis

The analysis was performed on binary pcap files logged by an unknown version of Snort Intrusion Detection System running in logging mode. The following files were downloaded from <http://isc.sans.org/logs/Raw>.

- 2002.10.16
- 2002.10.17
- 2002.10.18

To effectively analyze the traffic in the files, the three individual files are merged into one binary capture file using mergecap (figure 1). This provides a better view on trends across the network among hosts and attacks alike.

```
[analyst@recon ~]$ mergecap -w pcap 2002.10.16 2002.10.17 2002.10.18
```

Figure 1 – ‘mergecap -w [output file] [input file(s)]’

A network security analysis requires an understanding of hacker motives and methods. Additionally, to assess critical assets and their vulnerabilities, a fundamental knowledge of network and system architecture is essential. This includes technologies and behaviors. Attacks against organizations target a variety of platforms, from network devices, to servers and workstations. They can be both structured (deliberately targeted) and unstructured (scripted tools/worms). To effectively analyze the University’s network, it is essential to be cognizant of this fact, and diligently learn the network and its capacities, which includes topology, architecture, and technology.

The first step in the analysis is to break down the topology to get a feel for the network layout. Tethereal¹ is initially used to view the contents of the pcap file, including the date, time and types of traffic. Tethereal provides fast results via command line options – ideal for analysts – to get insight into the network and its systems.

```
[analyst@recon ~]$ tethereal -nr pcap -t ad
1 2002-11-15 16:26:47.606507 255.255.255.255 -> 170.129.209.73 TCP
31337 > 515 [RST, ACK] Seq=0 Ack=0 Win=0 Len=3
2 2002-11-15 16:32:16.826507 64.28.86.231 -> 170.129.50.120 HTTP
Continuation
3 2002-11-15 16:41:53.666507 255.255.255.255 -> 170.129.146.14 TCP
31337 > 515 [RST, ACK] Seq=0 Ack=0 Win=0 Len=3
4 2002-11-15 16:52:05.286507 211.47.255.21 -> 170.129.156.144 TCP 40037
> 0 [SYN] Seq=0 Ack=0 Win=5840 Len=0 MSS=1460 WS=0
<snip> ----- truncated
3088 2002-11-18 05:44:59.736507 170.129.50.120 -> 64.154.80.50 HTTP
Continuation
3089 2002-11-18 05:45:22.156507 170.129.50.120 -> 64.154.80.50 HTTP
Continuation
3090 2002-11-18 05:45:22.416507 170.129.50.120 -> 64.154.80.50 HTTP
Continuation
3091 2002-11-18 05:45:48.656507 170.129.50.120 -> 64.154.80.49 HTTP
Continuation
```

Figure 2 – ‘tethereal -nr [file name] -t ad’ – first four and last four packets

¹ Tethereal – text version of Ethernet protocol analyzer

Tethereal syntax:

- n – suppress name resolution
- r *file* – read input from file (*pcap*)
- t *options* – timestamp, a for ‘absolute’, d for ‘with date’

This initial run through with tethereal gives an insight into the binary file, showing the type of traffic and the timestamp of the capture. The first item to notice is that the dates of the entries in the file show this to be a capture from November rather than October, as the file names allude to.

To continue enumerating the network layout, tcpdump² is used, adding the ‘e’ switch to print layer 2 information. Using awk, only the source and destination MAC addresses are filtered out. This produces two MAC addresses consistent throughout the capture (figure 3).

```
[analyst@recon ~]$ /usr/sbin/tcpdump -ner pcap | awk '{print $2,$4}' |
sort -u | uniq -u
reading from file pcap, link-type EN10MB (Ethernet)
00:00:0c:04:b2:33 00:03:e3:d9:26:c0,
00:03:e3:d9:26:c0 00:00:0c:04:b2:33,
```

Figure 3 – tcpdump output to awk

Tcpdump syntax:

- n – suppress name resolution
- e – print Ethernet headers
- r *file* – read in *file*

To glean more information on the devices, a search is conducted on the vendor id of the MAC address (the first 24 bytes) on the IEEE OUI web page (<http://standards.ieee.org/regauth/oui/index.shtml>)³.

Here are the results of your search through the public section of the IEEE Standards OUI database report for 00-03-e3:		
00-03-E3	(hex)	Cisco Systems, Inc.
0003E3	(base 16)	Cisco Systems, Inc. 170 West Tasman Dr. San Jose CA 95134 UNITED STATES
Here are the results of your search through the public section of the IEEE Standards OUI database report for 00-00-0c:		
00-00-0C	(hex)	CISCO SYSTEMS, INC.
00000C	(base 16)	CISCO SYSTEMS, INC. 170 WEST TASMAN DRIVE SAN JOSE CA 95134-1706

Figure 4 – IEEE OUI Results

² Tcpdump – popular network/protocol analyzer

³ IEEE OUI and Company_id Assignments

So now it is evident that the IDS is sitting between two Cisco routing devices. Speculating on possibilities, this implementation could be between the border router facing the ISP and the perimeter router (or firewall) facing the internal network, but this can not be concluded just yet. Figure 5 depicts the sensor's location.

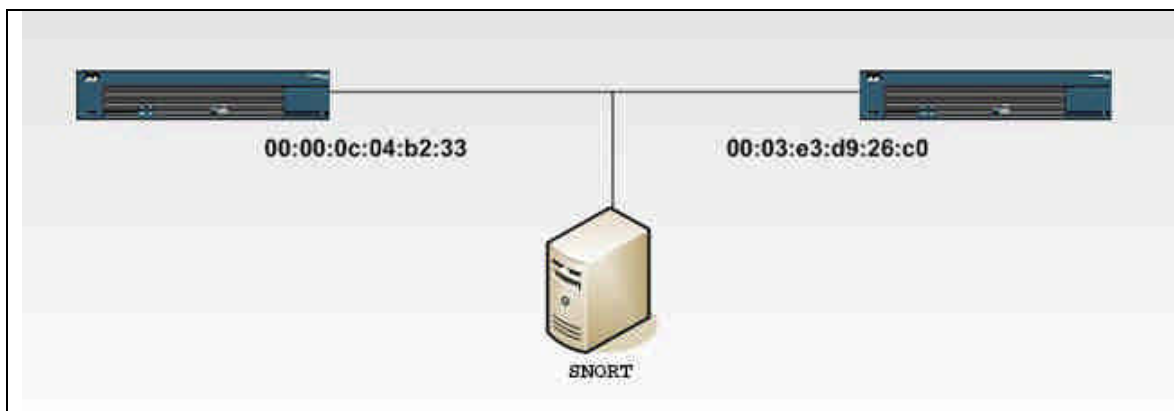


Figure 5 – Snort IDS sitting between two Cisco routing devices

To determine more about the setup, it is important to determine which device of the two is the gateway to the LAN and which device is the gateway to the ISP. Tethereal is used again to help determine the flow of traffic, starting with the device at Ethernet address 00:03:e3:d9:26:c0. Figure 6 shows the tethereal command checking the source device with MAC address 00:03:e3:d9:26:c0. Awk takes the output of tethereal, strips it down, and prints the source addresses associated with egress traffic on that device. Piping the output to 'sort -u' and 'uniq -u' sorts the IP addresses and shows only IPs which are unique. The output is a wide range of IP address space.

```
[analyst@recon ~]$ tethereal -nr pcap eth.src == 00:03:e3:d9:26:c0 |
awk '{print $3}' |sort -u | uniq -u
12.235.101.66
128.167.120.13
129.94.6.30
130.65.152.46
142.166.56.130
153.33.24.3
161.69.201.238
163.15.105.152
163.20.176.1
163.22.229.253
163.23.238.9
163.24.239.8
164.109.62.87
165.154.7.2
168.191.214.120
<snip>
```

Figure 6 – tethereal filter on source MAC 00:03:e3:d9:26:c0, awk showing source IPs

Next, tethereal is run again, filtering this time for the destination IP addresses. Figure 7 shows the output from this filter.

```
[analyst@recon ~]$ tethereal -nr pcap eth.src == 00:03:e3:d9:26:c0 |
awk '{print $5}' |sort -u | uniq -u
UNI.NET.100.206
UNI.NET.100.236
UNI.NET.100.243
UNI.NET.100.51
UNI.NET.10.221
UNI.NET.103.221
UNI.NET.104.249
UNI.NET.106.120
UNI.NET.106.86
UNI.NET.107.3
UNI.NET.107.88
UNI.NET.108.132
UNI.NET.108.46
UNI.NET.109.179
UNI.NET.111.141
UNI.NET.111.203
<snip>
```

Figure 7 – similar filter with awk printing the destination addresses

The output of this elicits IP space from what appears to be a Class B network, which is modified (first two octets) as UNI.NET throughout the rest of the analysis. The consistency produced by this last command implies that this is the internal LAN, which is validated by checking the other MAC address using the same filters (figures 8 and 9).

```
[analyst@recon ~]$ tethereal -nr pcap eth.src == 00:00:0c:04:b2:33 |
awk '{print $3}' |sort -u | uniq -u
```

Figure 8 - tethereal filter on 00:00:0c:04:b2:33 to display source IPs

```
[analyst@recon ~]$ tethereal -nr pcap eth.src == 00:00:0c:04:b2:33 |
awk '{print $5}' |sort -u | uniq -u
```

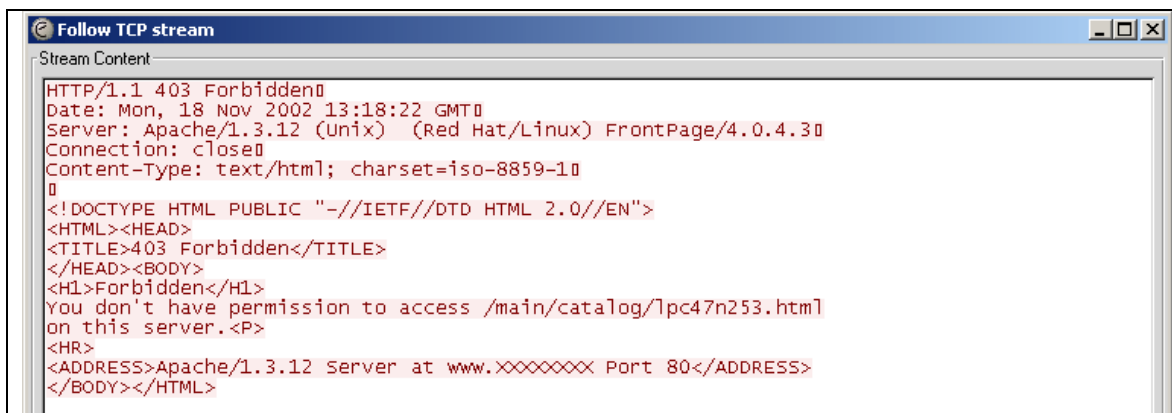
Figure 9 - same filter using '{print \$5}' to display destination IPs

It appears that the Cisco device with the MAC address 00:03:e3:d9:26:c0 is the external facing device (router). The internal network is attached to the Cisco device with MAC address 00:00:0c:04:b2:33, either another router, or a firewall. Since the egress traffic from this is limited to primarily http, it could very well be a firewall performing at minimum basic packet filtering on limited traffic.

More analysis on the file shows that outbound traffic originates from two internal hosts: UNI.NET.50.120 (making up the bulk of egress traffic) and UNI.NET.50.3 (two egress packets). Ethereal is used next to filter on both hosts to see if more insight into each of the two systems can be attained.

The host at UNI.NET.50.3 sends a couple of packets – error messages, which provide some information. Looking at the two error messages sent (http

403), the host at UNI.NET.50.3 appears to be a web server; Apache version 1.3.12 hosted on Red Hat Linux (figure 10).



```

Follow TCP stream
Stream Content
HTTP/1.1 403 Forbidden
Date: Mon, 18 Nov 2002 13:18:22 GMT
Server: Apache/1.3.12 (Unix) (Red Hat/Linux) FrontPage/4.0.4.3
Connection: close
Content-Type: text/html; charset=iso-8859-1
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>403 Forbidden</TITLE>
</HEAD><BODY>
<H1>Forbidden</H1>
You don't have permission to access /main/catalog/lpc47n253.html
on this server.<P>
<HR>
<ADDRESS>Apache/1.3.12 server at www.XXXXXXXXXX Port 80</ADDRESS>
</BODY></HTML>

```

Figure 10 - Apache HTTP 404 Error message

Looking at traffic targeting the server, there are a few items that stand out. There are Code Red attempts – six to be exact – against this system, and several requests for miscellaneous data, mostly web form requests.

The Code Red attempts stand out because of the unique signature they have, with the string 'GET /default.ida?NNNNNN...'. These attempts are against a UNIX based server, and since Code Red is a worm that targets Microsoft's IIS web server (Cert Advisory CA-2001-19), they can be considered false alarms.

The web form requests primarily reference FormMail scripts; an html based email utility that allows a form to be filled out on a website, and then sent via email to the recipient. If this server is, in fact, functioning in this capacity it is likely that there is sensitive data being stored either on the server or through a trust relationship with a database server. Either way, the safeguard of this system will be a priority to the University.

Attention is then focused on the other IP, UNI.NET.50.120. Traffic to/from this IP is a mix of tcp/80 (http), tcp/6667 (IRC), tcp/1863 (Messenger), and tcp/7000. Owing 1801 of 1803 egress packets, this IP either belongs to a proxy server, or the perimeter firewall is running Network Address Translation (NAT) using UNI.NET.50.120 as the public facing IP. Figure 11 shows both possibilities.

Continuing to look through the packets, more interesting items are noted. First of all, packets with a source address of 255.255.255.255 (ip broadcast)- these odd packets also have a source port of 31337 – something worth noting. Also, some of the http packets have 'cmd.exe' in the payload – definitely something to look at. Overall, there are several different protocols traversing the wire. These include scattered SMB packets, proxy requests, RPC requests, primarily http. Most of these are unidirectional packets, traffic coming in as requests, but no return traffic (or at least not seen in the capture). There are also inbound packets from a server that seem to be serving as a web proxy cache server (figure 12 shows a link graph with relations between some of the hosts).

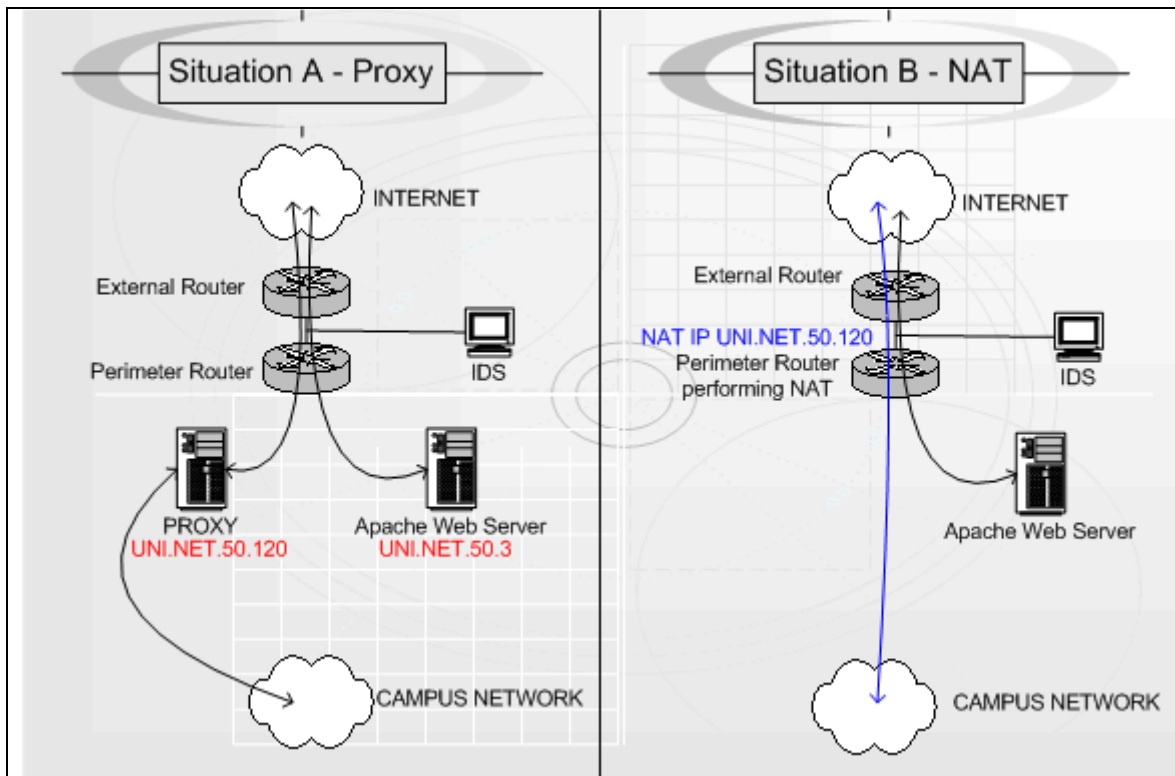


Figure 11 – Possible network configurations; Proxy versus NAT

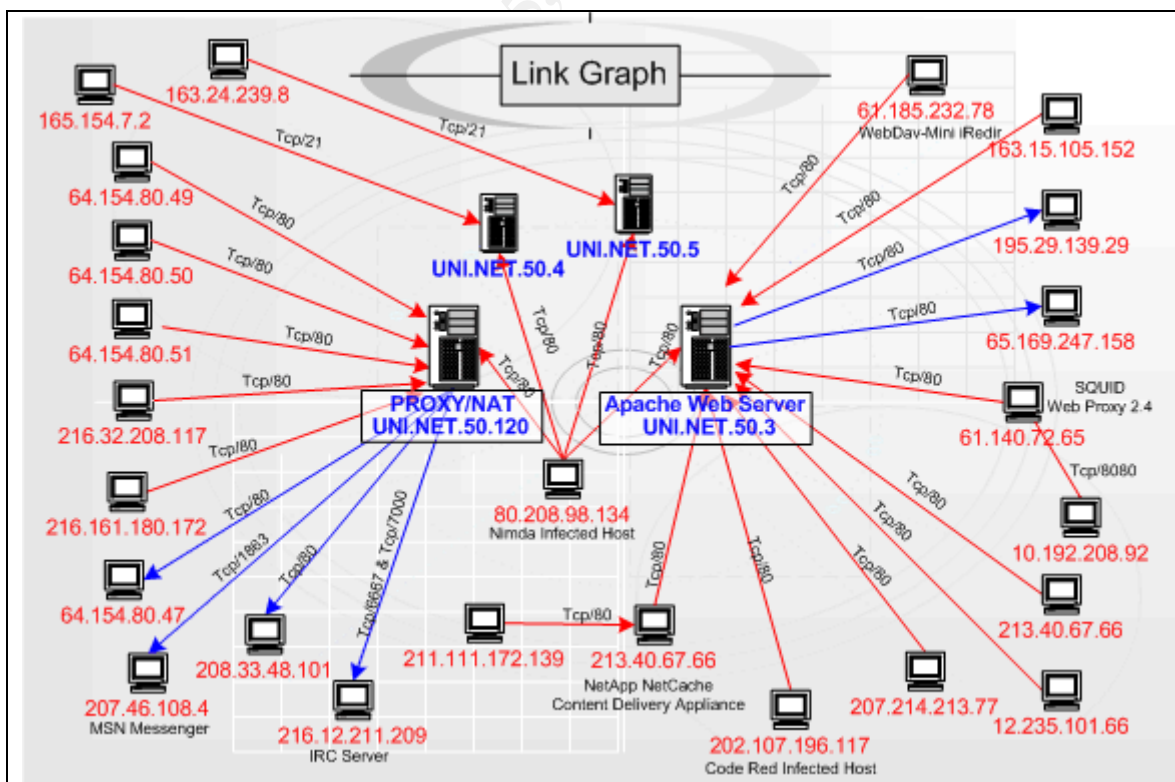


Figure 12 - Link graph showing traffic hitting high priority University targets

The next step is to run through the pcap file to view anomalous traffic seen on the network, using the popular open source IDS – Snort (figure 13).

```
[root@recon ~]# /usr/sbin/snort -r pcap -k none -c
/etc/snort/snort.conf
```

Figure 13 - Snort is run on the pcap file

Snort syntax:

- *r pcap* – read input from *pcap*
- *k option* – checksum mode, the option ‘none’ turns off checksum validation since the binary files do not contain original information.
- *c file* – use the specified snort configuration file

Snort performs an initialization, reading configuration from snort.conf (primary configuration file), which includes network parameters, variables and other custom information. This is where the customization of Snort is configured. After initialization, Snort produces a summary of the traffic, as figure 14 shows. Of 2,999 packets, there are 718 alerts, 16 unique.

```
<snip>
Snort processed 2999 packets.
Action Stats:
ALERTS: 718
LOGGED: 718
PASSED: 0
<snip>
```

Figure 14 - Summary statistics from Snort

With a high number of alerts, it helps to have an analyst console to view the data. Chapter 9 of Network Intrusion Detection, 2nd edition (Northcutt, Novak), reinforces this idea and lists some benefits:

- *Better false positive management*
- *Display filters*
- *Ability to mark events that have been analyzed*
- *Ability to drill down*
- *Correlation*
- *Better reporting*

ACID (Analysis Console for Intrusion Databases) provides a graphical user interface that taps into the MySQL database where Snort is configured to store alerts. The ACID console shows the alerts and statistical information of the file in an easy-to-use console.

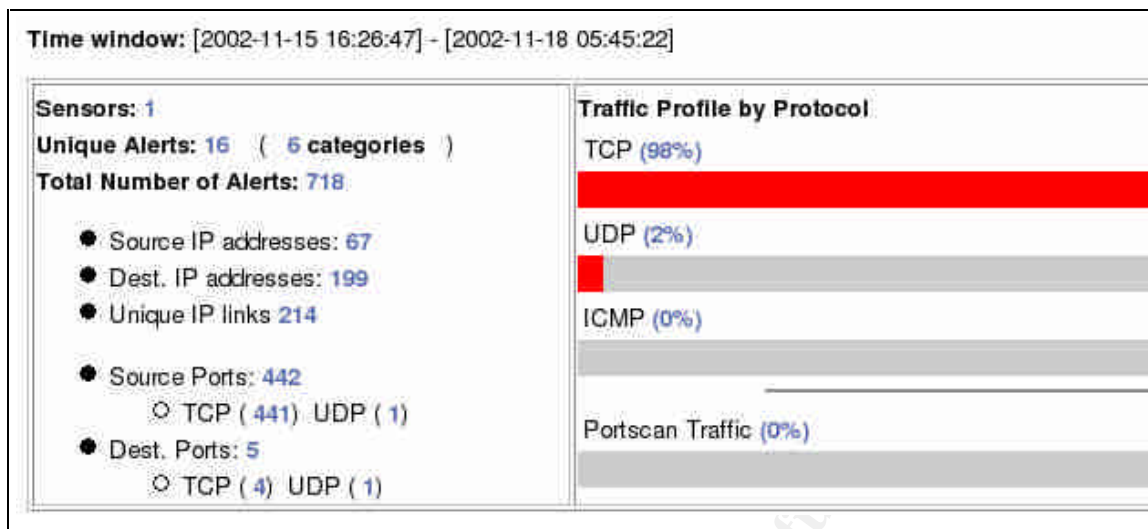


Figure 15 – ACID displays alert statistics

ACID confirms the total number of alerts; 718 (16 unique alerts, 6 unique categories). The first thing to notice is the listed classification of alerts. Figure 16 displays the ACID classification summary. This provides an opportunity to see the types of attacks being seen on the University Campus network. ACID lists the classification of each attack, along with some information on the number of alerts (number of occurrences of each), how many different signatures are listed under each of the classifications, and the source/destination IP information.

< Classification >	< Total # >	< Sensor # >	< Signatures >	< Src. Addr. >	< Dest. Addr. >
unclassified	420 (58%)	1	7	19	47
misc-activity	242 (34%)	1	3	8	114
bad-unknown	36 (5%)	1	2	36	36
shellcode-detect	2 (0%)	1	1	2	2
attempted-recon	2 (0%)	1	2	1	1
rpc-portmap-decode	16 (2%)	1	1	1	1

Figure 16 – The six classifications of alerts, the total number of each, and related information.

This visual summary can be useful in quickly triaging the events, especially when dealing with an improperly configured IDS sensor alerting indiscriminately. It is important to point out that this is not a conclusive response method, as alerts will be firing on legitimate packets. In prioritizing events with non-legitimate traffic, it is important to remember that a successful reconnaissance attack is not as high of a priority as a successful exploit (root access or privilege escalation) attack...unless of course the access attack proves ineffective (Windows exploit attack on a UNIX server) and the reconnaissance attack elicits critical information to the attacker. This could be a prelude to an exploit based attack.

Table 1 shows the alerts by name followed by occurrences that were detected on the University's network.

Summary of Alerts:

<i>Alert Name</i>	<i>Occurrences</i>
BACKDOOR Q access	93
BAD-TRAFFIC tcp port 0 traffic	137
(http_inspect) NON-RFC HTTP DELIMITER	17
(http_inspect) WEBROOT DIRECTORY TRAVERSAL	76
(snort_decoder) WARNING: TCP Data Offset is less than 5!	2
BAD-TRAFFIC same SRC/DST	35
(http_inspect) OVERSIZE REQUEST-URI DIRECTORY	107
MISC Tiny Fragments	1
(http_inspect) BARE BYTE UNICODE ENCODING	205
SHELLCODE x86 0xEB0C NOOP	2
SCAN FIN	1
(snort_decoder): Tcp Options found with bad lengths	1
BAD-TRAFFIC ip reserved bit set	12
(http_inspect) DOUBLE DECODING ATTACK	5
RPC portmap mountd request UDP	16
(http_inspect) IIS UNICODE CODEPOINT ENCODING	8

Table 1 - Summary of Alerts

Sixteen different alerts are listed in the table, followed by the number of occurrences. The alerts that have parenthesis are generated by Snort's preprocessors, while the others are detected by signatures.

Looking at the alerts, some appear as less of a priority than others. The priorities here, with little known about the hosts residing on the internal network, are alerts that signify an exploit, root access, or privilege escalation. The Backdoor Q access, Web root directory traversal, and Shellcode alerts stand out as such alerts.

Detect 1: BACKDOOR Q access

```
[**] [1:184:6] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
11/17-17:11:56.686507 255.255.255.255:31337 -> UNI.NET.166.76:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0 Ack: 0x0 Win: 0x0 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]
```

Description of detect:

Q is a remote admin tool commonly associated as a backdoor utility, such as Sub Seven, or Back Orifice. Q is described by its author, Mixer, as a “Remote shell and admin tool with strong encryption”. Although Q was primarily built for UNIX, it could be compiled to run on the operating system of choice. This program has the ability to run in a ‘stealth mode’, encrypting packets between the attacker and victim. The stealth mode capability limits the need to establish a valid TCP connection, which may provide the opportunity to bypass simple packet filtering firewalls, or intrusion detection systems.

This has been given a candidate CVE entry - CVE# CAN-1999-0660 which is still under review. The description for this CVE entry is “A hacker utility, back door, or Trojan Horse is installed on a system, e.g. NetBus, Back Orifice, Rootkit, etc.”

Reason this detect was selected:

With a source address of 255.255.255.255, and source port of 31337, this packet is at the top of the list of odd packets. The address, first of all, is not a valid address; its primary use is as a destination address in a BOOTP packet. The source port of 31337 is interesting also, as 31337 is hacker speak for *elite* (31337 = ‘e/leet’)⁴. Q is a program that, when run, can provide remote root access to the target. Unsolicited access, especially root access to unknowing victim systems, should be mitigated at all costs, as that is the primary goal of intrusion detection/prevention. The traffic pattern that matches the alert was seen across the wire ninety-three times, essentially ninety three potential victims. With just one being too many, this alert needs to be investigated.

Detect generated by:

This alert (figure 17) was generated by Snort Intrusion Detection System, version 2.2.0. The following backdoor alert was triggered on ninety three events.

⁴ The term ‘eleet’ or ‘leet’ is slang for hackers who are highly skilled (elite), usually written as ‘31337’ or ‘1337’.

```
[**] [1:184:6] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
11/17-17:11:56.686507 255.255.255.255:31337 -> UNI.NET.166.76:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A**R** Seq: 0x0 Ack: 0x0 Win: 0x0 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]
```

Figure 17- Backdoor Q alert

These alerts were triggered by a rule (figure 18) that incorporates traditional signature features with preprocessor features. One of Snort's features is the ability to use preprocessor output for signature input. The purpose of this is to normalize data (traffic) prior to alerting, thus reducing false positives. The alert header attempts to match an address, and the alert options add preprocessor functions (*flow:stateless* is part of the Flow preprocessor module).

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

Figure 18 - Snort Q Signature

Breaking down the alert into four components, the features of the signature are a little clearer.

```
alert tcp 255.255.255.0/24 any(1) -> $HOME_NET any(2) (msg:"BACKDOOR Q
access"(3); dsize:>1; flags:A+(4); flow:stateless; reference:arachnids,203; classtype:misc-
activity; sid:184; rev:6;)
```

- | | |
|---|---|
| 1 | Alert, on any TCP traffic from the 255.255.255.0 network, any source port |
| 2 | Destined to the internal network, any destination port |
| 3 | Message to display to console |
| 4 | The 'ACK' flag is set (to indicate a response) |

Figure 19 - Breakdown of the 'Backdoor Q access' signature

The packets that violate the rule have a source address of 255.255.255.255, are destined for the internal network, and have 'ACK' flag set. The flags in the TCP header are located in the 14th byte (figure 21), so a filter using tcpdump (figure 20) on packets with a source belonging to the 255.255.255.0 network and with the ack flag set would look as follows:

```
[analyst@recon ~]$ /usr/sbin/tcpdump -nnvX -r pcap net 255.255.255.0/24 && (tcp[13]
& 0x05)
reading from file pcap, link-type EN10MB (Ethernet)
16:26:47.606507 IP (tos 0x0, ttl 14, id 0, offset 0, flags [none], proto 6, length: 43)
255.255.255.255.31337 > UNI.NET.209.73.515: R [tcp sum ok] 0:3(3) ack 0 win 0 [RST
cko]
    0x0000: 4500 002b 0000 0000 0e06 3103 ffff ffff E..+.....1.....
    0x0010: aa81 d149 7a69 0203 0000 0000 0000 0000 ...Izi.....
    0x0020: 5014 0000 e52a 0000 636b 6f00 0000    P....*..cko...
<snip>
```

Figure 20 - TCP filter

```
tcpdump -nnvX -r pcap net 255.255.255.0/24 && (tcp[13] & 0x05)
```

Tcpdump syntax is as follows:

- - nn, no name resolution, including ports
- - v, verbose
- - X print ASCII
- - r pcap, read in from file pcap
- net 255.255.255.0/24 – any source or destination host belonging to that network
- tcp[13] – check the 14th byte (starting from 0) of the tcp header
- 0x05 – match the fifth place (from right), which is the ack flag field

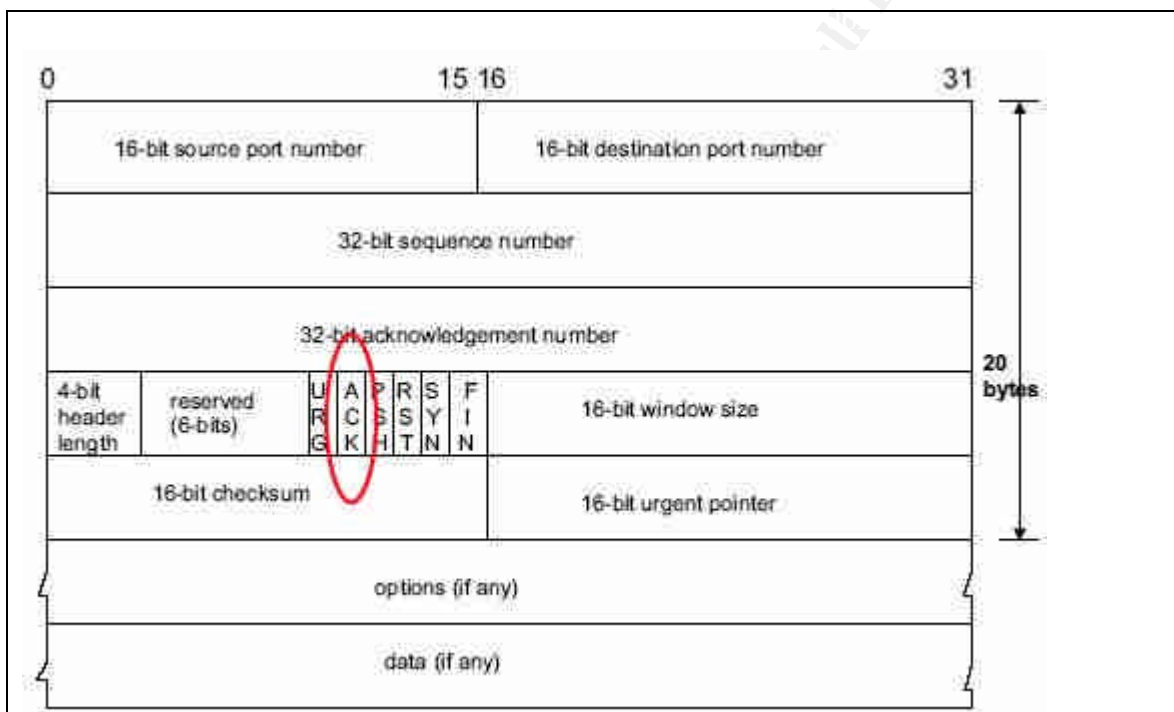


Figure 21 - TCP Header showing the ACK flag position

Probability the source address was spoofed:

There is a high probability that this source address is spoofed. This address stands out as an IP broadcast address, or *limited broadcast address* (Stevens, 217), usually seen as the destination address in client request packets while bootstrapping (RFC 951, BOOTP); therefore it is likely a spoofed address.

Attack mechanism:

This attack targets victim hosts running the Q server (qd). As a backdoor program, this may be part of a rootkit⁵ allowing an attacker running the Q client

⁵ A rootkit is one or more tools used by a hacker that is uploaded to a compromised system to hide evidence of a system compromise, and to hide active hacker processes trojaned programs etc.

(qs) to spawn a remote shell with root access on the target. According to the readme file provided with Q version 2.4:

“...only qd and qs from the same compilation will work with each other, because of hard-coded random authenticity ID's generated by mkpasswd.”

This alludes to the fact that if any of the hosts were infected, then they would have to be targeted by a client of the same build. This is a high possibility, given the ubiquity of the internet, and an equal ubiquity of attack delivery mechanisms (worms etc).

The purpose of Q is multifold. In the hands of malicious users, it can serve as a method to compromise targeted computers. With root access to a system (or multiple systems), new attacks can be launched from ‘owned’⁶, or compromised computers. This is a popular method for several types of attacks, including DDoS (Distributed Denial of Service) and relay attacks.

In the packets seen in the Q attacks, the RST and ACK flags were set. It is unlikely that the targeted systems would reply to RST packets, since RST is the signal to abort or abruptly cancel a connection. If the targeted systems were the initiators of the traffic, then we might see a retransmission of packets, but this was not the case.

Correlations:

This detect was analyzed in depth by Les Gordon (GCIAC practical) in 2002, where he actually analyzed versions 0.9, 1.0, 2.0, and the most recent, 2.4. ArachNIDS lists the backdoor as IDS203, “Trojan-Active-Q-TCP”, referring to Q as a Trojan. Computer Associates’ *Pest Patrol* also lists Q as a Trojan as well on the website where it states:

“A Remote Administration Tool, or RAT, is a Trojan that when run, provides an attacker with the capability of remotely controlling a machine via a “client” in the attacker's machine, and a “server” in the victim's machine.”

The Common Vulnerabilities and Exposures entry CAN-1999-0660 (under review) classifies this as “A hacker utility, back door, or Trojan Horse is installed on a system, e.g. NetBus, Back Orifice, Rootkit, etc.”

Evidence of active targeting:

This attack targeted ninety-three victims in three days; this does not appear to be a structured attack, hence no active targeting. Trojans, for the most part, are attacks that start with a scan looking for vulnerable systems. The vulnerable system, or victim, will reply back to the sender of the scan indicating that it is infected. As the attacker scans IP addresses and finds vulnerable hosts, it is likely attacks will be sent to those hosts to exploit the systems. These events appear to be the initial scans. Furthermore, targeted attacks are usually executed in stealthy manners to bypass intrusion detection systems and these attacks are anything but stealthy.

⁶ The term ‘owned’, commonly written as ‘0wn3d’, is used to refer to hacked systems.

Severity

$$\text{Severity} = (2+5) - (4+3) = 0$$

$$\text{Severity} = (\text{criticality} + \text{lethality}) - (\text{system countermeasures} + \text{network countermeasures})$$

Criticality: 2 – Although it is necessary to be vigilant here, the targets in this attack appear as part of a scan, and do not elicit any response to the attacker.

Lethality: 5 – Q provides root access to remote victims, the most lethal of attacks.

System Countermeasures: 4 – the targets in this case did not appear to be vulnerable to this attack, and did not reply to the RST packets.

Network Countermeasures: 3 – The routers should not be routing packets containing a limited broadcast address, unless the environment is using bootp/dhcp relay. There was, however, no indicator that the packets were successful in entering the private network, so the perimeter device could have silently dropped the packets.

© SANS Institute 2005, Author retains full rights.

Detect 2: WEBROOT DIRECTORY TRAVERSAL

```
[**] [119:18:1] (http_inspect) WEBROOT DIRECTORY TRAVERSAL [**]
11/15-19:25:58.326507 211.87.212.36:4061 -> UNI.NET.93.33:80
TCP TTL:100 TOS:0x0 ID:56820 IpLen:20 DgmLen:136 DF
***AP*** Seq: 0x9241C916 Ack: 0x1235 Win: 0x4470 TcpLen: 20
```

Description of detect:

A directory traversal (*file system traversal*) is an attack that involves the use of a web browser to execute commands on a remote web server. The goal is to bypass the built-in security features of the server that restrict users to a web root folder. The ability to break out of the web root folder can lead to full system access. These types of attacks have been used to deface websites, steal information, and exploit trust relationships with database servers to propagate the theft of private data (credit card numbers etc).

The Snort Signature Database⁷ describes this attack as follows:

“This event is generated when the http_inspect pre-processor detects an attempt to escape the root directory of a web server by an attacker using a directory traversal technique.”

The US-CERT lists this as Vulnerability Note VU#111677, with direct reference to Microsoft’s TechNet Security Bulletin MS00-78 “Microsoft IIS 4.0 / 5.0 vulnerable to directory traversal via extended unicode in url (MS00-078)”. Two entries from the CERT Coordination Center stem from MS00-78; Cert Incident Note IN-2001-09: “Code Red II: Another Worm Exploiting Buffer Overflow In IIS Indexing Service DLL’, and CERT Advisory CA-2001-26 Nimda Worm. Finally, the CVE entry for this is CVE-2000-0884.

Reason this detect was selected:

Due to the ramifications of a successful directory traversal attack on a web-server, it is critical to investigate potential attacks. Although this could very well be a scripted attack, or a network worm, as seen with Code Red variants and subsequent Nimda variants, it could also be a targeted attack.

Detect generated by:

This detect was generated by Snort Intrusion Detection System, version 2.2.0. The alert was not by triggered by a signature, but by one of Snort preprocessors. Figure 22 shows the alert that was generated.

```
[**] [119:18:1] (http_inspect) WEBROOT DIRECTORY TRAVERSAL [**]
11/15-19:25:58.326507 211.87.212.36:4061 -> UNI.NET.93.33:80
TCP TTL:100 TOS:0x0 ID:56820 IpLen:20 DgmLen:136 DF
***AP*** Seq: 0x9241C916 Ack: 0x1235 Win: 0x4470 TcpLen: 20
```

Figure 22- http_inspect preprocessor alert

⁷ <http://www.snort.org/snort-db/sid.html?sid=119:18>

The preprocessor options are configured in Snort.conf, the primary configuration file for Snort. There are two options in snort.conf as seen in figure 23.

```
preprocessor http_inspect: global \
    iis_unicode_map unicode.map 1252

preprocessor http_inspect_server: server default \
    profile all ports { 80 8080 8180 } oversized_dir_length 500
```

Figure 23 - http_inspect preprocessor entries in snort.conf

A note about preprocessors:

Snort uses preprocessors to make up for the shortcomings of attack signatures. Preprocessors increase detection abilities beyond simple pattern-matching signatures into more functional detection capabilities, including protocol anomaly detection. This is especially helpful in complicated situations where a generalized attack signature results in too many false positives, yet a specific attack signature can miss attacks, resulting in numerous false negatives. The preprocessor is invoked after Snort's decoder has broken the packet into fields and before the packet is matched against known attack signatures.

Decoder → Preprocessor → Signature engine

The http_inspect preprocessor, developed by Daniel Roelker of Sourcefire, has the built-in capabilities to detect a number of http anomalies, including IDS evasion techniques as well as web based http attacks.

Probability the source address was spoofed:

Given the nature of attack, it is not likely that the source address is spoofed. The traffic appears to be worm attacks from infected source addresses scanning for vulnerable hosts to propagate.

Attack mechanism:

Directory Traversals have different attack mechanisms as both structured and unstructured attacks. In this case, the attack mechanism appears to be related to network worms carrying a payload designed to exploit vulnerable web server technologies, specifically Microsoft IIS vulnerabilities.

Looking at the data of the packet that triggered the first alert, there is an http GET request for cmd.exe. This raises a flag, since cmd.exe is an administrative utility for Windows systems. To look for the host initiating this traffic, ngrep is used (figure 24), searching for packets matching 80.208.98.134:

```
[analyst@recon ~]$ ngrep -I pcap cmd.exe host 80.208.98.134
input: pcap
filter: ip and ( host 80.208.98.134 )
match: cmd.exe
#
T 80.208.98.134:2317 -> UNI.NET.50.3:80 [AP]
  GET /scripts/..%5c%5c../winnt/system32/cmd.exe?/c+dir..ir..
#
T 80.208.98.134:2319 -> UNI.NET.50.4:80 [AP]
  GET /scripts/..%5c%5c../winnt/system32/cmd.exe?/c+dir..ir..
```



```

#
T 80.208.98.134:2323 -> UNI.NET.50.5:80 [AP]
GET /scripts/../../../../winnt/system32/cmd.exe?/c+dir..ir..
#
T 80.208.98.134:2339 -> UNI.NET.50.14:80 [AP]
GET /scripts/../../../../winnt/system32/cmd.exe?/c+dir..ir..
#
T 80.208.98.134:2341 -> UNI.NET.50.15:80 [AP]
GET /scripts/../../../../winnt/system32/cmd.exe?/c+dir..ir..
#
T 80.208.98.134:2353 -> UNI.NET.50.21:80 [AP]
GET /scripts/../../../../winnt/system32/cmd.exe?/c+dir..ir..
#
T 80.208.98.134:2359 -> UNI.NET.50.23:80 [AP]
GET /scripts/../../../../winnt/system32/cmd.exe?/c+dir..ir..
#
T 80.208.98.134:2368 -> UNI.NET.50.28:80 [AP]
GET /scripts/../../../../winnt/system32/cmd.exe?/c+dir..ir..
#
T 80.208.98.134:2359 -> UNI.NET.50.23:80 [AP]
GET /scripts/../../../../winnt/system32/cmd.exe?/c+dir.dir.
#
T 80.208.98.134:2317 -> UNI.NET.50.3:80 [AP]
GET /scripts/../../../../winnt/system32/cmd.exe?/c+dir.dir.
#
T 80.208.98.134:2353 -> UNI.NET.50.21:80 [AP]
GET /scripts/../../../../winnt/system32/cmd.exe?/c+dir.dir.
#
T 80.208.98.134:2552 -> UNI.NET.50.120:80 [AP]
GET /scripts/../../../../winnt/system32/cmd.exe?/c+dir..ir..
exit

```

Figure 24 - ngrep shows numerous matching packets

Ethereal is used to view the packet data (figure 25), as it provides a quick method of reconstructing TCP streams, which is extremely useful in analysis of large pcap files. The first attack contains the following payload:

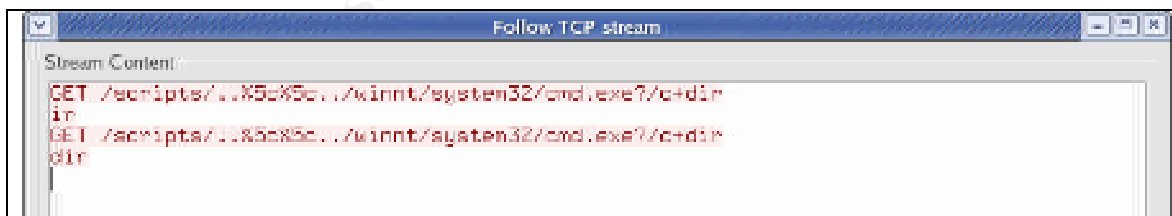


Figure 25 - Part of a screen capture showing Ethereal's TCP stream reassembly

There are four source IP addresses that triggered these alerts. Filtering on the attacker's IP address of 80.208.98.134, there are 12 entries across 9 unique IP destinations residing on the University LAN. The time deltas between the packets are small, indicating that this is more of a scan than a targeted attack. Further research on CERT reveals that this is likely to be a Nimda worm scan. The packets each have the 'P' (tcp 'push') and 'ACK' (tcp 'acknowledgement') flags set, which are set during the exchange of data (after the initial TCP connection). However there were no previous connection indications, and this

trace shows a unidirectional data flow which is an indication of an infected host scanning for vulnerable targets (figure 26).

```
[analyst@recon ~]$ /usr/sbin/tcpdump -nn -r pcap host 80.208.98.134 -
tttt
reading from file pcap, link-type EN10MB (Ethernet)
2002-11-17 19:58:41.176507 IP 80.208.98.134.2317 > UNI.NET.50.3.80: P
545794100:545794159(59) ack 3022993836 win 64620
2002-11-17 19:58:41.196507 IP 80.208.98.134.2319 > UNI.NET.50.4.80: P
545882922:545882981(59) ack 3022275420 win 64620
2002-11-17 19:58:41.246507 IP 80.208.98.134.2323 > UNI.NET.50.5.80: P
546112382:546112441(59) ack 3029926665 win 64620
2002-11-17 19:58:41.366507 IP 80.208.98.134.2339 > UNI.NET.50.14.80: P
546882327:546882386(59) ack 1470430696 win 64620
2002-11-17 19:58:41.376507 IP 80.208.98.134.2341 > UNI.NET.50.15.80: P
546975249:546975308(59) ack 2981604386 win 64620
2002-11-17 19:58:41.386507 IP 80.208.98.134.2353 > UNI.NET.50.21.80: P
547536443:547536502(59) ack 3030427244 win 64620
2002-11-17 19:58:41.396507 IP 80.208.98.134.2359 > UNI.NET.50.23.80: P
547811586:547811645(59) ack 3028618036 win 64620
2002-11-17 19:58:41.416507 IP 80.208.98.134.2368 > UNI.NET.50.28.80: P
548253445:548253504(59) ack 3303634675 win 64620
2002-11-17 19:58:41.596507 IP 80.208.98.134.2359 > UNI.NET.50.23.80: P
59:117(58) ack 2873 win 0
2002-11-17 19:58:41.616507 IP 80.208.98.134.2317 > UNI.NET.50.3.80: P
59:117(58) ack 5627073 win 0
2002-11-17 19:58:41.966507 IP 80.208.98.134.2353 > UNI.NET.50.21.80: P
59:117(58) ack 2873 win 0
2002-11-17 19:58:45.636507 IP 80.208.98.134.2552 > UNI.NET.50.120.80: P
558165330:558165389(59) ack 3681321655 win 64620
```

Figure 26 - tcpdump output for host 80.208.98.134

Looking at the next attacker IP (211.87.212.36), a similar pattern exists, with a similar payload, and both 'P' and 'ACK' flags set (figure 27). With twenty-two packets in this attack, there are only three targeted hosts. Two hosts are hit with eight packets, and one is hit with six. As with the previous attack, the time deltas on each victim are fast.

```
[analyst@recon ~]$ /usr/sbin/tcpdump -nn -r pcap host 211.87.212.36 -
tttt
reading from file pcap, link-type EN10MB (Ethernet)

2002-11-16 03:25:58.326507 IP 211.87.212.36.4061 > UNI.NET.93.33.80: P
2453784854:2453784950(96) ack 4661 win 17520
2002-11-16 03:25:58.426507 IP 211.87.212.36.4067 > UNI.NET.93.33.80: P
2454439935:2454440052(117) ack 4661 win 17520
2002-11-16 03:25:58.446507 IP 211.87.212.36.4084 > UNI.NET.93.33.80: P
2454736513:2454736630(117) ack 4661 win 17520
2002-11-16 03:25:58.536507 IP 211.87.212.36.4090 > UNI.NET.93.33.80: P
2455432209:2455432354(145) ack 4661 win 17520
2002-11-16 03:26:06.266507 IP 211.87.212.36.1426 > UNI.NET.93.33.80: P
2520477553:2520477651(98) ack 4661 win 17520
2002-11-16 03:26:06.326507 IP 211.87.212.36.1432 > UNI.NET.93.33.80: P
2521173002:2521173098(96) ack 4661 win 17520
2002-11-16 03:26:06.366507 IP 211.87.212.36.1445 > UNI.NET.93.33.80: P
```

```

2521409592:2521409692(100) ack 4661 win 17520
2002-11-16 03:26:06.446507 IP 211.87.212.36.1449 > UNI.NET.93.33.80: P
2522073604:2522073700(96) ack 4661 win 17520

2002-11-16 22:36:23.186507 IP 211.87.212.36.2268 > UNI.NET.113.11.80: P
3518162550:3518162646(96) ack 4661 win 17520
2002-11-16 22:36:23.186507 IP 211.87.212.36.2271 > UNI.NET.113.11.80: P
3518275784:3518275901(117) ack 4661 win 17520
2002-11-16 22:36:23.196507 IP 211.87.212.36.2273 > UNI.NET.113.11.80: P
3518445137:3518445254(117) ack 4661 win 17520
2002-11-16 22:36:23.216507 IP 211.87.212.36.2297 > UNI.NET.113.11.80: P
3519651267:3519651365(98) ack 4661 win 17520
2002-11-16 22:36:23.216507 IP 211.87.212.36.2302 > UNI.NET.113.11.80: P
3519886404:3519886500(96) ack 4661 win 17520
2002-11-16 22:36:23.226507 IP 211.87.212.36.2313 > UNI.NET.113.11.80: P
3520345658:3520345754(96) ack 4661 win 17520

2002-11-17 02:27:37.476507 IP 211.87.212.36.1393 > UNI.NET.130.226.80:
P 3747232701:3747232797(96) ack 4661 win 17520
2002-11-17 02:27:37.506507 IP 211.87.212.36.1412 > UNI.NET.130.226.80:
P 3748143833:3748143950(117) ack 4661 win 17520
2002-11-17 02:27:37.516507 IP 211.87.212.36.1429 > UNI.NET.130.226.80:
P 3749009714:3749009831(117) ack 4661 win 17520
2002-11-17 02:27:37.556507 IP 211.87.212.36.1450 > UNI.NET.130.226.80:
<snip>

```

Figure 27 - tcpdump output for host 211.87.212.36

Although very similar to the first attack, these are somewhat different. Repeated attempts on the same victim and a slightly different payload (figure 28).

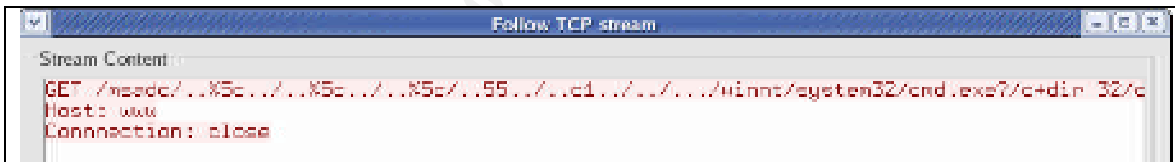


Figure 28 – Ethereal TCP reassembly showing HTTP payload: IIS worm

According to CERT Advisory CA-2001-26:

“The scanning activity of the Nimda worm produces the following log entries for any web server listing on port 80/tcp:

```

GET /scripts/root.exe?/c+dir
GET /MSADC/root.exe?/c+dir
GET /c/winnt/system32/cmd.exe?/c+dir
GET /d/winnt/system32/cmd.exe?/c+dir
GET /scripts/../../../../%5c/../../../../winnt/system32/cmd.exe?/c+dir
GET /_vti_bin/../../../../%5c/../../../../winnt/system32/cmd.exe?/c+dir
GET /_mem_bin/../../../../%5c/../../../../winnt/system32/cmd.exe?/c+dir
GET
/masdc/../../../../%5c/../../../../%5c/../../../../xc1\x1c../../../../xc1\x1c../../../../xc1\x1c/win
nt/system32/cmd.exe?/c+dir
GET /scripts/../../../../xc1\x1c../../../../winnt/system32/cmd.exe?/c+dir
GET /scripts/../../../../xc0/../../../../winnt/system32/cmd.exe?/c+dir
GET /scripts/../../../../xc0\xaf../../../../winnt/system32/cmd.exe?/c+dir
GET /scripts/../../../../xc1\x9c../../../../winnt/system32/cmd.exe?/c+dir

```

```
GET /scripts/..%35c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%35c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%5c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%2f../winnt/system32/cmd.exe?/c+dir
```

Note: The first four entries in these sample logs denote attempts to connect to the backdoor left by Code Red II, while the remaining log entries are examples of exploit attempts for the Directory Traversal vulnerability.”

All four addresses have the same characteristics. These packets match up to the CERT advisory showing that they are likely Nimda infected hosts scanning for vulnerable targets.

Correlations:

This detect originated from a vulnerability noted in Microsoft’s IIS web server product. It was discovered by Rain Forest Puppy, and Microsoft subsequently released Security Bulletin (MS00-78), “Patch Available for 'Web Server Folder Traversal' Vulnerability”. It has been given CVE entry CVE-2000-0884.

Evidence of active targeting:

Given the primary means of propagating, these packets are not actively targeting the victims. Instead, they are randomly scanning the victims for vulnerable services.

Severity

Severity = (4+5) – (4+3) = 1

Severity = (criticality + lethality) – (system countermeasures + network countermeasures)

Criticality: 4 – Although there was no indication of a successful attack, at least one of the targets in this attack is a critical server. This is due to its function as a web server, and other trust relationships it has with other internal machines.

Lethality: 5 – This attack, if successful, can lead to a full directory traversal resulting in a full system compromise.

System Countermeasures: 4 – Since the victims in these attacks did not appear to respond, it is assumed they were not vulnerable to the specific attack and/or had anti-virus software in place to protect them.

Network Countermeasures: 3 – Although filtering does not appear to be in place on the internet facing device, the perimeter device appears to be filtering for specific addresses.

Detect 3: SHELLCODE x86 0xEB0C NOOP

```
[**] [1:1424:6] SHELLCODE x86 0xEB0C NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
11/16-08:41:40.176507 163.24.239.8:2377 -> UNI.NET.50.5:21
TCP TTL:44 TOS:0x0 ID:35386 IpLen:20 DgmLen:560 DF
***AP*** Seq: 0xAB7BA6BD Ack: 0xA5C3AABB Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 4675704 5583989
```

Description of detect:

This attack involves the use of an exploit to compromise a vulnerable system, or service, to gain root privileges. This particular attack is targeting a vulnerable FTP service. If FTP is running as root (uid 0), the success of this attack will allow the attacker to gain root privileges (the privileges of the targeted service). According to Snort's description of this event;

“This event is generated when suspicious shell code is detected. Many buffer overflow attacks contain large numbers of NOOP instructions to pad out the request. Other attacks contain specific shell code sequences directed at certain applications or services.”

Shellcode is essentially code written with the sole purpose of eliciting a remote shell on the target system. As The Shellcoder's Handbook states (Kozio, 3); “Shellcode is defined as a set of instructions injected and then executed by an executable program. Shellcode is used to directly manipulate registers and the function of a program, so it must be written in hexadecimal opcodes”

The term ‘NOOP’ (no-op), also called ‘NOP’, stands for ‘no operation’ and refers to assembly instructions that instruct a computer's processor to do nothing. ‘NOPping’ is commonly used in assembly programming to pad memory locations around a set of actual useful instructions. This is beneficial to the attacker, when the vulnerability being exploited exists at a specific point in memory. Trying to be that accurate is a game of hit and miss, so padding the exploit with NOP instructions gives a broader target. The CPU will return the stack pointer to the attacker's code, execute the NOP code (do nothing) until the actual exploit code is reached. This is also called a NOP sled, because the stack pointer just slides across the NOP commands until the desired code is executed.

Reason this detect was selected:

Shellcode attacks are potentially lethal by nature, as shellcode attacks were designed to give a remote attacker a root shell on the targeted system. Without any prior knowledge of the network systems and the types of software being run, it is impossible to know what is vulnerable. Therefore, a shellcode attack needs to be scrutinized.

Detect generated by:

This detect was generated again by Snort IDS version 2.2 on a Shellcode signature. The following alert:

```
[**] [1:1424:6] SHELLCODE x86 0xEB0C NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
11/16-08:41:40.176507 163.24.239.8:2377 -> UNI.NET.50.5:21
TCP TTL:44 TOS:0x0 ID:35386 IpLen:20 DgmLen:560 DF
***AP*** Seq: 0xAB7BA6BD Ack: 0xA5C3AABB Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 4675704 5583989
```

Figure 29 - Shellcode Alert

was generated by the following signature:

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any
(msg:"SHELLCODE x86 0xEB0C NOOP"; content:"|EB 0C EB 0C EB 0C EB 0C EB
0C EB 0C EB 0C EB 0C|"; classtype:shellcode-detect; sid:1424; rev:6;)
```

Figure 30 - Shellcode Signature

The signature alerts on repeated ‘EB 0C’ characters in the payload of the datagram. The ‘content’ section of the signature looks for at least eight (8) iterations of the ‘EB 0C’ characters, as such an array would be an indicator of a NOP sled (figure 31).

```
alert ip1 $EXTERNAL_NET2 $SHELLCODE_PORTS3 -> $HOME_NET4 any5
(msg:"SHELLCODE x86 0xEB0C NOOP"; content:"|EB 0C EB 0C EB
0C EB 0C EB 0C EB 0C EB 0C EB 0C5|"; classtype:shellcode-
detect; sid:1424; rev:6;)
```

- | | |
|----------|---|
| 1 | Alert, on any IP traffic (transport layer not regarded here) |
| 2 | From outside the internal (local) network (defined by \$EXTERNAL_NET) |
| 3 | From source ports defined in snort.conf to any destination port |
| 4 | Destined to internal (local) network (defined by \$HOME_NET) |
| 5 | Payload data containing NOOP commands (‘EB 0C’) |

Figure 31 - Breakdown of the Shellcode alert

The snort.conf file used in this analysis does not give specific definition to the internal and external networks, as the full network layout was not known. The definition for both was set to ‘any’, meaning any source to any destination. This particular alert did not give cause to either the external or internal network addresses, Snort alerted solely on the packets crossing the wire that contained the data specified with the ‘content’ keyword *EB 0C*.

Probability the source address was spoofed:

This attack has a low probability that the address is spoofed. TCP based attacks require that a valid connection be made, which requires a response from both sides. Since both the source and destination have to send and received packets to establish the connection, it follows that the IP addresses are valid

Attack mechanism:

This attack uses a NOP sled to employ an FTP exploit. To understand the attack mechanism, it is necessary to determine exactly what exploit is being used. For this, a detailed analysis of the payload of each of the packets is

needed. There are two separate attacks, each with a unique source and destination (figure 32).

ID	Signature	Timestamp	Source Address	Dest. Address	Layer 4 Proto
#0-(1-292)	[snort] SHELLCODE x86 0xEB0C NOOP	2002-11-17 04:54:27	165.154.7.2:1982	170.129.50.4:21	TCP
#1-(1-144)	[snort] SHELLCODE x86 0xEB0C NOOP	2002-11-16 07:41:40	163.24.239.8:2377	170.129.50.5:21	TCP

Figure 32 - Two Shellcode Alerts

Looking at the two attacks, it is evident that each sent three packets. Using tcpdump, the first packet in the attack is extracted. Figure 33 shows the command and subsequent output.

```
[analyst@recon ~]$ /usr/sbin/tcpdump -nnr pcap -xXvv tcp[2:2]=0x15 -q -S -tttt
reading from file pcap, link-type EN10MB (Ethernet)
2002-11-16 15:41:40.176507 IP (tos 0x0, ttl 44, id 35386, offset 0, flags [DF], proto 6,
length: 560) 163.24.239.8.2377 > 170.129.50.5.21: tcp 508
0x0000: 0000 0c04 b233 0003 e3d9 26c0 0800 4500 .....3....&...E.
0x0010: 0230 8a3a 4000 2c06 53e6 a318 ef08 aa81 .0.:@.,.S.....
0x0020: 3205 0949 0015 ab7b a6bd a5c3 aabb 8018 2..I...{.....
0x0030: 7d78 dd79 0000 0101 080a 0047 5878 0055 }x.y.....GXx.U
0x0040: 3475 4357 4420 3030 3030 3030 3030 3030 4uCWd.0000000000
0x0050: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x0060: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x0070: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x0080: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x0090: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x00a0: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x00b0: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x00c0: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x00d0: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x00e0: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x00f0: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x0100: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x0110: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x0120: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x0130: 3030 3030 3030 3030 3030 3030 3030 3030 0000000000000000
0x0140: 3030 3030 3030 f0fc 4031 0708 985f 0808 000000..@1..._..
0x0150: eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c .....
0x0160: eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c .....
0x0170: eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c .....
0x0180: eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c .....
0x0190: eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c .....
0x01a0: eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c .....
0x01b0: eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c .....
0x01c0: eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c .....
0x01d0: eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c .....
0x01e0: eb0c eb0c eb0c eb0c eb0c eb0c eb0c eb0c .....
0x01f0: eb0c eb0c eb0c 9090 9090 9090 9090 9090 .....
0x0200: 9090 31db 43b8 0b74 510b 2d01 0101 0150 ..1.C..tQ.-...P
0x0210: 89e1 6a04 5889 c2cd 80eb 0e31 dbf7 e3fe ..j.X.....1....
0x0220: ca59 6a03 58cd 80eb 05e8 ed0a ca59 6a03 .Yj.X.....Yj.
0x0230: 58cd 80eb 05e8 edff ffff ffff ff0a X.....
```

Figure 33 - Packet 1 of 3 of the first attack

The tcp[2:2] locates the destination port in the TCP header, starting at byte 3, higher order nibble, spanning 2 bytes. Setting this equal to '0x15' evaluates the value for port 21 (FTP) hexadecimal. The '-S' keeps the sequence

numbers absolute, since it is essential to look for multiple packets of the same session between the source and destination. The '-tttt' option keeps the date in the original format.

The content of this packet reveals some notable findings. First of all, the NOP commands that fired this alert 'eb0c' are seen repeatedly in the packet. Also note the trailing '9090' references that follow, these are also NOP commands. Looking at the payload also reveals the 'CWD' ftp command. This packet reveals information, but to be conclusive, further exploration of the other packets is necessary. The next two packets in this attack (figure 34) are smaller, but similar:

```

2002-11-16 15:41:40.796507 IP (tos 0x0, ttl 44, id 35478, offset 0, flags [DF], proto 6,
length: 68) 163.24.239.8.2377 > 170.129.50.5.21: tcp 16
0x0000: 0000 0c04 b233 0003 e3d9 26c0 0800 4500 .....3....&...E.
0x0010: 0044 8a96 4000 2c06 5576 a318 ef08 aa81 .D..@.,.Uv.....
0x0020: 3205 0949 0015 ab7b a8b9 a5c3 acc4 8018 2..I...{.....
0x0030: 7c70 caf3 0000 0101 080a 0047 58be 0055 |p.....GX..U
0x0040: 34b9 4357 4420 7e2f 7b2e 2c2e 2c2e 2c2e 4.CWD.~/{.....
0x0050: 7d0a                                     }.

2002-11-16 15:41:55.306507 IP (tos 0x0, ttl 44, id 37517, offset 0, flags [DF], proto 6,
length: 59) 163.24.239.8.2377 > 170.129.50.5.21: tcp 7
0x0000: 0000 0c04 b233 0003 e3d9 26c0 0800 4500 .....3....&...E.
0x0010: 003b 928d 4000 2c06 4d88 a318 ef08 aa81 ;;..@.,.M.....
0x0020: 3205 0949 0015 ab7b a921 a5c3 adfb 8018 2..I...{.!......
0x0030: 7c70 3072 0000 0101 080a 0047 5e6f 0055 |pOr.....G^o.U
0x0040: 3a6b 4357 4420 7e7b 0a                   :kCWD.~{.

```

Figure 34 - Packets 2 & 3 of first attack

Ethereal's TCP stream reassembly function was used to see the payload of the three consecutive packets (figure 35).

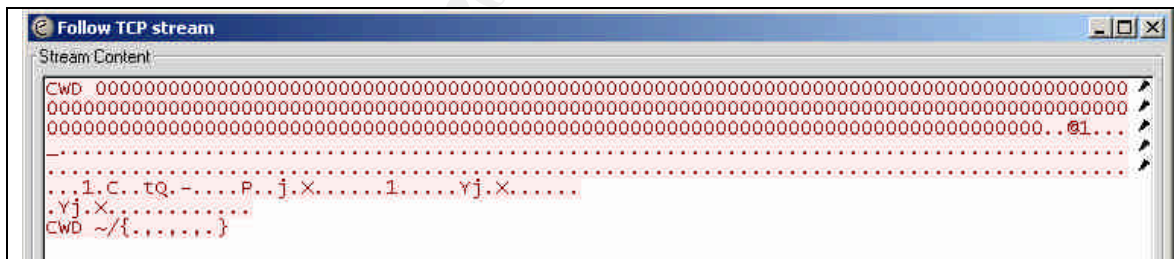


Figure 35 - Contents of packets

Looking at Ben Allen's analysis of a similar attack, the signature was alerting on a similar FTPd CWD attack. The packets he analyzed in his practical on page 12 were almost identical to the initial ones seen in these attacks:

- TTL = 44
- ID = 35386
- Len = 560
- Size = 508

There are multiple known FTP exploits that could elicit this type of packet, two popular ones being the Vermilion FTPd attack, and the WU-FTPd attack. The Vermilion attack is one of the popular FTP exploits, however, looking at the code

provided by USSR Labs, the 'CWD' command should come after the 'user' and 'pass' requests, which is not seen in the two attacks. Also, in the packet, CWD is succeeded by zeros, and according to the source code, it's succeeded as follows:

```
"'CWD aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'".
```

The WU-FTPd attack (also popular), has several versions and exploits. The shellcode in the WU-FTPd attacks use NOP sleds, however the NOPs is a little different, implementing the '9090' method of 'NOP'ing as follows

```
unsigned char  x86_wrx[] =
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

    "\x31\xdb\x43\xb8\x0b\x74\x51\x0b\x2d\x01\x01\x01"
    "\x01\x50\x89\xe1\x6a\x04\x58\x89\xc2\xcd\x80\xeb"
    "\x0e\x31\xdb\xf7\xe3\xfe\xca\x59\x6a\x03\x58\xcd"
    "\x80\xeb\x05\xe8\xed\xff\xff\xff";
```

According to CVE entry 2001-0550, there are implementations of Wu-FTPd that do not correctly handle file name globbing. The characteristic that entices a deeper investigation is the characters "~{" which are passed as command parameters to 'CWD'. This was seen in the second and third packets in each attack.

Another detail to point out is the time between packets. The first two packets have an almost identical time stamp – 15:41:40, the third packet is off by 00:00:15, which is still relatively quick. It can be concluded that the attack is not specific to just one, but more of a general FTPd scan, looking for systems running vulnerable FTP implementations.

Correlations:

This was analyzed by Ben Allen in his practical from April 2004, where packets appeared to be similar to the packets in this analysis. He alluded to the fact that his attack targeted Vermilion FTPd. This was again analyzed by Kam Hung Ng in May 2004 leading to the conclusion that the similar attack was targeting Wu FTPd.

This was given a CVE entry of CVE-2001-0550, and a US Cert Vulnerability Note VU#886083.

Evidence of active targeting:

If this is, in fact, a scripted utility or scan, then active targeting can be discounted. Scans like this are popular among unskilled attackers who download utilities and perform scans on subnets in hope of finding a vulnerable system. With only two packets in this trace, there is limited information to use to determine targeting. Based on the analysis, it appears that active targeting can be ruled out.

Severity

$$\text{Severity} = (4+5) - (3+3) = 3$$

$$\text{Severity} = (\text{criticality} + \text{lethality}) - (\text{system countermeasures} + \text{network countermeasures})$$

Criticality: 4 – The systems targeted in this attack have given little indication that they are running FTP processes, however the subnet UNI.NET.50.0/24 correspond to other hosts that are running Web services, and should be an indicator to keep an eye on.

Lethality: 5 – This attack, if successful, can lead to root access of the target system.

System Countermeasures: 3 – The targeted systems did not respond to the attackers in the capture, however given the nature of attacks, it can be assumed this system is running vulnerable software.

Network Countermeasures: 3 – Although filtering does not appear to be in place on the internet facing device, the perimeter device appears to be filtering for specific addresses.

© SANS Institute 2005, Author retains full rights.

Network Statistics

Using tcpdstat, a concise snapshot of information from the binary capture file is shown (figure 36).

```
[root@recon analyst]# tcpdstat pcap

DumpFile: pcap
FileSize: 2.17MB
Id: 200211151626
StartTime: Fri Nov 15 16:26:47 2002
EndTime: Mon Nov 18 05:45:48 2002
TotalTime: 220741.05 seconds
TotalCapSize: 2.13MB CapLen: 1514 bytes
# of packets: 3091 (2.57MB)
AvgRate: 372.44bps stddev:1758.85 PeakRate: 169.57Kbps

### IP flow (unique src/dst pair) Information ###
# of flows: 560 (avg. 5.52 pkts/flow)
Top 10 big flow size (bytes/total in %):
 15.3% 8.7% 7.3% 6.9% 6.2% 5.9% 5.7% 5.2% 3.1% 3.0%

### IP address Information ###
# of IPv4 addresses: 586
Top 10 bandwidth usage (bytes/total in %):
 91.9% 15.3% 8.7% 7.3% 6.9% 6.2% 5.9% 5.7% 5.2% 3.1%
### Packet Size Distribution (including MAC headers) ###
<<<<
 [ 32- 63]:    596
 [ 64- 127]:   300
 [ 128- 255]:  330
 [ 256- 511]:  284
 [ 512- 1023]: 402
 [ 1024- 2047]: 898
 [ 2048- 4095]: 250
 [ 4096- 8191]: 31
>>>>

### Protocol Breakdown ###
<<<<
  protocol      packets      bytes      bytes/pkt
-----
[0] total      3091 (100.00%) 2697979 (100.00%) 872.85
[1] ip         3091 (100.00%) 2697979 (100.00%) 872.85
[2] tcp        3040 ( 98.35%) 2694311 ( 99.86%) 886.29
[3] ftp         6 ( 0.19%)    1458 ( 0.05%)   243.00
```

[3]	dns	1 (0.03%)	60 (0.00%)	60.00
[3]	http(s)	385 (12.46%)	380991 (14.12%)	989.59
[3]	http(c)	1874 (60.63%)	2233055 (82.77%)	1191.60
[3]	netb-se	12 (0.39%)	1356 (0.05%)	113.00
[3]	socks	107 (3.46%)	6448 (0.24%)	60.26
[3]	squid	107 (3.46%)	6434 (0.24%)	60.13
[3]	irc6667	1 (0.03%)	75 (0.00%)	75.00
[3]	irc7000	12 (0.39%)	900 (0.03%)	75.00
[3]	http-a	122 (3.95%)	7366 (0.27%)	60.38
[3]	other	400 (12.94%)	55388 (2.05%)	138.47
[2]	udp	16 (0.52%)	1568 (0.06%)	98.00
[3]	sunrpc	16 (0.52%)	1568 (0.06%)	98.00
[2]	igmp	35 (1.13%)	2100 (0.08%)	60.00
[2]	frag	94 (3.04%)	119400 (4.43%)	1270.21
>>>>				

Figure 36 - tcpdstat provides statistical network information

Top Five Talkers

To define the top talkers, the number of packets were transmitted was used as criteria. To do that, of Ethereal's statistical features was used (figure 37). Based on the number of packets sent, the 5 top 'talkers' are as follows

UNI.NET.50.120 202.108.254.200 64.125.138.190 255.255.255.255 202.108.254.204

Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
UNI.NET.50.120	1642	1359635	1371	989358	271	370277
202.108.254.200	221	13260	221	13260	0	0
64.125.138.190	112	140174	112	140174	0	0
255.255.255.255	93	5580	93	5580	0	0
202.108.254.204	86	5160	86	5160	0	0

Figure 37 - Ethereal produces helpful statistical information

Top Five Targeted services or ports

The top 5 targeted ports, based on the number of occurrences, are as follows:

Port	Protocol	Occurrences
80	Tcp	420
0	Tcp	137
515	Tcp	93
111	Udp	16
21	Tcp	2

Figure 38 - Top 5 targeted ports

Three most suspicious external source addresses

Although there were quite a few 'suspicious' external IP addresses, there were a few hosts that stood out as being more suspicious than others. Two of these were proxy systems, and the third was an IRC server.

Table 2 lists the top three suspicious IP addresses.

IP	Function
61.140.72.65	Squid Web Proxy Cache v 2.4
213.40.67.66	NetApp NetCache Appliance
216.12.211.209	IRC Server

Table 2 - Three most suspicious external hosts

The two external proxies were chosen due to the nature of a proxy system. Attacks can be relayed through a proxy to the victim, eliminating traces of the original attacker. The IRC, or Internet Relay Chat, server was also chosen due to the nature of IRC communication. IRC is a popular method of communication used by hackers to express information regarding new exploits found or existing exploits used to target victims of choice.

The first IP (61.140.72.65) appeared to be a UNIX system running Squid Web Proxy v 2.4 based on the http payload. The operating system is assumed to be UNIX, since Squid only runs on UNIX based systems. The timestamp in the http requests (GET and POST) shows a January date, although the packets show a November timestamp, which could imply an improperly configured server. An APNIC search shows that the following host is based in China:

inetnum:	61.140.72.64 - 61.140.72.79
netname:	GUANGZHOU-ELEC-COMM-CENTER
descr:	GUANGZHOU ELECTRIC COMMUNICATION CENTER
country:	CN
admin-c:	LW240-AP
tech-c:	LW240-AP
mnt-by:	MAINT-CHINANET-GD
changed:	ipadm@gddc.com.cn 20010703
status:	ASSIGNED NON-PORTABLE
source:	APNIC
changed:	hm-changed@apnic.net 20020827

Figure 39 - Squid Proxy host look up via APNIC

The host at IP 213.40.67.66 is also functioning as a proxy. Based on the http requests, it appears to be running NetApp's NetCache content delivery appliance.

According to RIPE, this device resides in the UK (see figure 36).

inetnum:	213.40.0.0 - 213.40.255.255
role:	The Internexus Group
address:	Indigo House, Time Technology Park
address:	Blackburn Road, Simonstone
address:	Burnley, BB12 7NQ, UK
phone:	+44 1282 681 320
e-mail:	operations@internexusgroup.co.uk
trouble:	abuse@supanet.com
admin-c:	GA1249-RIPE
tech-c:	GA1249-RIPE
nic-hdl:	IG464-RIPE

```

remarks:      Please use abuse@internexusgroup.co.uk to report Abuse
changed:      stephen.bailey@internexusgroup.co.uk 20040315
source:       RIPE

```

Figure 40 - RIPE information on 213.40.67.66

The last IP, 216.12.211.209, appears to be an IRC server residing in Texas (figure 37).

```

OrgName:      Everyones Internet, Inc.
OrgID:        EVRY
Address:      2600 Southwest Freeway
Address:      Suite 500
City:         Houston
StateProv:    TX
PostalCode:   77098
Country:      US

NetRange:     216.12.192.0 - 216.12.223.255
CIDR:         216.12.192.0/19
NetName:      EVRY-BLK-4
NetHandle:    NET-216-12-192-0-1
Parent:       NET-216-0-0-0-0
NetType:      Direct Allocation
NameServer:   NS1.EV1.NET
NameServer:   NS2.EV1.NET
Comment:      ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:      2000-04-24
Updated:      2000-06-29

```

Figure 41 - ARIN output on 216.12.211.209

Thirteen packets were sent to this server on ports 6667 and 7000. Each packet has the string "NICK [SVCDP]-XDCC-35" which, according to Marcus Wu's posting to the University of Stuttgart's RUS CERT intrusion list, is an attempt to switch nick handles to [SVCDP]-XDCC-35. The XDCC refers to an IRC bot (or multiple) that use the DCC protocol for file sharing (Wikipedia). Therefore it can be concluded that internal University hosts are file sharing via IRC.

Correlations

In this analysis of the 'Backdoor Q access' detect, Les Gordon's practical was instrumental. Gordon conducted a thorough analysis of the known and available versions of Q, and seemed to be reflected in a SANS Intrusion Detection FAQ article regarding the Q Trojan. Detailed information was pulled from the Q 'readme' file as well as some of the Q installation files.

Information was used from idsresearch.org on Web based attacks. Other information was correlated with Snort's primary configuration file (snort.conf) and the main book used as a reference for Snort, Snort 2.1 Second Edition (Beale).

GCIA practicals from Ben Allen and Kam Hung Ng were used during the analysis of the Shellcode FTP attacks. Both provide good insight, as well as a cause and effect analysis to ensure related to the FTP attacks.

Insights into internal machines

There were at least two servers functioning in the capacity of web server. Both UNI.NET.50.3 and UNI.NET.50.120 were noted sending and receiving web traffic. The UNI.NET.50.0/24 subnet appears to be a web layer, given most of the web communication (including the FTP attack) was targeting this subnet. The host at UNI.NET.50.16 was targeted using tcp/139 indicating possible Microsoft NetBIOS services running on the network, however this proved inconclusive due to the lack of bi-directional traffic.

Defensive recommendations

Based on the analysis of the network traffic files, there are several points of concern, in which defensive measures need to be taken. Starting from the outside and working in, the following areas can be enhanced to protect both the network and the systems on the network.

- External (border) Network Defenses
- DMZ
- Perimeter Network Defenses
- Internal Network Defenses
- System Defenses

External (border) network defenses: The first step in protecting the internal local area network (LAN) is controlling the traffic hitting the border router/firewall. Applying access control lists, ingress/egress filtering, and rate limiting on the external router will prevent attacks such as Denial of Service (DoS) attacks that affect internal machines and network bandwidth alike.

DMZ: Moving public facing servers, such as web servers, to the DMZ will help prevent direct access into the internal network.

Perimeter network defenses: A stateful inspection firewall should be placed as a border to the internal network. Since this is a University, and a wide variety of traffic may be permitted on it, outbound connections will be permitted on a stateful level, limiting inbound traffic to authorized packets only.

Internal network defenses: Internal VLANS (virtual local area networks) should be employed to further segment internal hosts, adding once again another layer of security. This would be beneficial to the University in separating student activity from administrative activity. An additional sensor should be placed on the internal network, monitoring internal VLANS, and should be tuned to mitigate false positives.

System defenses: All critical systems on the network need to have antivirus software installed and need to be regularly updated to current patch levels, and current anti-virus definitions. Host based intrusion prevention should be considered for deployment on critical servers (web, email, database) for an added layer of security that is less intrusive to other hosts.

Part III – Analysis Process

The primary system used for the analysis was a Linux system running Fedora Core 2. Tcpdump and Ethereal, two of the most popular network analyzers used by analysts, were both installed and used. Each of the two had benefits, and hence, each was used for different purposes.

Ethereal's 'follow tcp stream' function was essential in reconstructing the packets in the file, to see the human readable version of the payload. Ethereal's command line version, Tethereal, accepted the same filter syntax as Ethereal, so it proved quite useful.

When using BPFs (Berkeley Packet Filters), tcpdump was the tool of choice. With its speed and flexibility, tcpdump was able to quickly sort through packets, producing the desired results. Tcpdump also has built in macros, which saved time by providing quick filters by using keywords.

The meat of the analysis relied upon Snort Intrusion Detection System. The version installed originally was 2.1.3, however upgraded to 2.2 during the course of the analysis. Snort was configured to log to MySQL, which stored all of the alerts and related information. To extract the alerts and to keep everything somewhat organized, ACID was used, running on Apache, using PHP to query the MySQL database.

Additional software included ngrep, and p0f. Ngrep has the ability to parse both live traffic and pcap files, accepting BPF filters, as well as regular expression matching. The regular expression matching was effective in this analysis, permitting simple strings as arguments.

P0f is a passive fingerprinting utility, and like ngrep, can be used both on live traffic and pcap files alike. P0f was used on the pcap file used in the analysis to attempt to fingerprint hosts for an OS type, and although successful on some hosts, was not able to fingerprint all.

Software	Version
Linux, Fedora Core 2	default
Snort	2.2
MySQL	4
ACID	0.9.3
Apache	1.3
P0f	1.8.3
Ngrep	1.40.1, rev 1.23

Figure 42 – Analysis software and versions used.

A separate computer was used for the compilation of the practical itself. Originally, Open Office 11 was used on the Fedora Core 2 system; however compatibility issues kept arising when viewing the assignment on other computers. It was eventually replaced by a Windows XP Professional, running Office XP, and Visio 2002.

Works Cited

- Allen, Ben. "Task-Optimized Operating Systems for Intrusion Detection." GIAC Certified Intrusion Analysts (GCIA) May 2004 GIAC
- "Analysis Console for Intrusion Databases (ACID)." Analysis Console for Intrusion Databases. 0.9.6b23. 08 Jan 2003 SourceForge.net <http://acidlab.sourceforge.net/>
- "American Registry for Internet Numbers." ARIN.net November 2004 ARIN <http://www.arin.net>
- "CAN-1999-0660 (under review)." Common Vulnerabilities and Exposures CVE 1999 <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660>
- "CAN-2001-0550." Common Vulnerabilities and Exposures CVE 2002 <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0550>
- Caswell, Brian et al. Snort 2.1 Intrusion Detection, 2nd Edition Rockland: Syngress, 2004
- "CERT Advisory CA-2001-26 Nimda Worm." CERT.org. 19 Sep 2001. Cert Coordination Center. October 2004 <http://www.cert.org/advisories/CA-2001-26.html>
- "CERT Incident Note IN-2001-09." CERT.org 17 Jan 2002. Cert Coordination Center. October 2004 http://www.cert.org/incident_notes/IN-2001-09.html
- Dittrich, Dave. "Tools written/modified by Dave Dittrich." September 2004 <http://staff.washington.edu/dittrich/talks/core02/tools/tools.html>
- Gordon, Les M. "Intrusion Analysis – The Director’s Cut." GIAC Certified Intrusion Analysts (GCIA) November 22, 2002 GIAC http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc
- "IEEE OUI and Company_id Assignments." IEEE 29 October 2004 IEEE <http://standards.ieee.org/regauth/oui/index.shtml>
- Kozio, Jack, et al. The Shellcoder’s Handbook, Discovering and Exploiting Security Holes. Indianapolis: Wiley, 2004

- “LOGS: GIAC GCIA Version 3.3 Practical (Marcus Wu).” RUS CERT January 16, 2003
 Universitat Stuttgart
<http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00120.html>
- “Microsoft Security Bulletin (MS00-78).” Microsoft TechNet 17 Oct 2000 Microsoft.com
 October, 2004 <http://www.microsoft.com/technet/security/bulletin/MS00-078.msp>
 .mixter security | .code” mixter security 2002 by Mixer <http://mixter.void.ru/code.html>
- Northcutt, Stephen, and Judy Novak. Network Intrusion Detection, An Analyst’s Handbook 2nd Edition. Indianapolis: New Riders, 2001
- “NSA/SNAC Router Security Configuration Guide.” NSA Version 1.1
http://www.nsa.gov/snac/routers/cisco_exec_sum.pdf
- Stevens, W. Richard TCP/IP Illustrated, Volume 1: The Protocols Indianapolis: Addison-Wesley
 1994
- “Snort Signature Database.” Snort August 2004 <http://www.snort.org/cgi-bin/done.cgi>
- “TCPDUMP public repository.” Tcpdump/libcap. 22 Jun 2004. tcpdump.org. August 2004
<http://www.tcpdump.org>
- “tethereal - The Ethereal Network Analyzer 0.10.7.” Ethereal September 2004
<http://www.ethereal.com/docs/man-pages/tethereal.1.html>
- “US-CERT Vulnerability Note VU#111677.” Microsoft IIS 4.0 / 5.0 vulnerable to directory traversal via extended unicode in url (MS00-078). December 04 2000 US-Cert. October 2004 <http://www.kb.cert.org/vuls/id/111677>
- “US-CERT Vulnerability Note VU#886083.” WU-FTPD does not properly handle file name globbing April 30, 2001 US-Cert October 2004 <http://www.kb.cert.org/vuls/id/886083>
- Roelker, Daniel J. “HTTP IDS Evasions Revisited.” IDSResearch.org :: about intrusion detection
 August 1, 2003 IDSResearch.org http://docs.idsresearch.org/http_ids_evasions.pdf
- “XDCC.” Wikipedia, The Free Encyclopedia November 24, 2004 Wikipedia
<http://en.wikipedia.org/wiki/XDCC>