



Interested in learning
more about security?

SANS Institute InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Simple Traffic Analysis With Ethereal

This paper describes how to use the Ethereal Display Filter to examine a capture log file. The data analyzed was recorded by port and the amount of packet traffic received. The attack patterns that emerged from the data analysis generally correspond with well published vulnerabilities from expected open ports on a server. Attackers also seem to have a variety of ways to get a server and/or firewall to acknowledge traffic and verify a potential target.

Copyright SANS Institute
Author Retains Full Rights

AD



MobileIron

EMM Strategy on the right track?
Know your security risks.

TAKE THE ASSESSMENT

Simple Traffic Analysis With Ethereal

GIAC Security Essentials
Certification (GSEC)
Practical Assignment
Version 1.4c

Option 1 - Research on Topics
in Information Security

Submitted by: <Neil Orlando>
Location: <SANS@Home>
March 16, 2005.

Paper Abstract: The use of Ethereal Display filters
to examine a capture log and see patterns in
attempted attacks against the server.

Table of Contents

Abstract/Summary	1
Introduction	1
The Setup	1
A Review and Use of Filters	2
Results of Viewing the Captured Data with Display Filters	4
Conclusion	12
References	14

List of Figures

Figure 1. Filter Tool Bar	2
Figure 2. TCP 135 NT RPC	5
Figure 3. TCP 445 Microsoft SMB	6
Figure 4. UDP 137 NetBIOS Name Server	10
Figure 5. Protocol Hierarchy Summary Graph	11

© SANS Institute 2000 - 2005, Author retains full rights.

Abstract/Summary

This paper describes how to use the Ethereal Display Filter to examine a capture log file. The data analyzed was recorded by port and the amount of packet traffic received. The attack patterns that emerged from the data analysis generally correspond with well published vulnerabilities from expected open ports on a server. Attackers also seem to have a variety of ways to get a server and/or firewall to acknowledge traffic and verify a potential target.

Introduction

The goal of this project was to review traffic to a specific IP address by examining a capture log file via Ethereal, identifying the ports an intruder tried to use to connect, and finding any backdoor programs that use those ports.

The Setup

An additional IP, 66.92.xx.xx, was aliased to a network interface card (NIC) of a functioning web server to make it multi-homed. Tcpcdump was run to capture the traffic to the IP address before the traffic hit the firewall and was dropped. This mechanism protected the existing web server that existed on another IP address. Nothing was running on IP 66.92.xx.xx during the test. In short, there were no reachable services.

The server was connected directly to the internet, and there were no routers, firewalls, or other packet filters in the way. In addition, no capture filters were used and the ISP did not block ports, so every packet captured was an actual packet that somebody tried sending to that IP address.

A total of 4297 packets were captured during an eight-day period:

Feb. 9, 2005, 21:52 hrs through Feb 18, 2005, 10:23 hrs

The machine from which these log files were captured ran Red Hat 9, kernel version 2.4.20-8.

Analysis was done on a Windows XP Professional box with an installation of Wincap 3.0 and Ethereal version 0.10.5.

A Review and Use of Filters

Ethereal uses filtering to help sort and find the data. Ethereal can use capture filters and display filters. The capture filters are used when logging data to a file for later analysis. “The capture filter syntax follows the same syntax that Tcpdump uses from the libpcap library. This is used on the command line or in the capture filter dialog box to capture certain types of traffic.”¹ This paper does not review capture filters, since the data examined for the test was already captured. The test used display filters to review the captured data and match traffic with specific protocols. Please note that display filters have a different command format from capture filters².

The following steps were used to load the existing capture file in order to view it.³

File – Open – change drives and directory to log capture file – Open

Display filters were accessed in two ways for this test:

- 1) Menu Bar - click Analyze → Display Filters. This brings up the Ethereal Display Filter dialog box
- 2) via the filter tool bar found near the top of Ethereal screen (note: older versions of Ethereal had the filter tool bar on the bottom of the screen).



Figure 1. Filter Tool Bar

- Click on button listed as Filter on the left hand side to bring up the Ethereal Display Filter dialog box; or,
- use the text box if you know the exact filter you want to use.
The left middle text box provides an area to enter or edit display filter strings... A syntax check of your filter string is done while you are typing. The background will turn red if you enter an incomplete or invalid string, and will become green when you enter a valid string. You can click on the pull down arrow to select a previously-entered filter string from a list. The entries in the pull down list will remain available even after a program restart.⁴

¹ Orebaugh, Angela. Ethereal Packet Sniffing. Syngress, 2004. Pg. 48.

² More information about capture filters can be found:
Sharpe, Richard. <http://www.ethereal.com/docs/user-guide-sp/#ChUseCaptureMenuSection>;
and,

Orebaugh, Angela. Ethereal Packet Sniffing. Syngress, 2004. Pg. 209.

³ Ethereal can read capture files from a variety of different tools. See the online documentation for more information on file formats: Sharpe, Richard. <http://www.ethereal.com/docs/user-guide-sp/#ChUseFilterToolbarSection>.

A new filter must be created when the display filter box is opened if this is the first instance of running display filters. The following example illustrates how to create a filter to see all HTTP traffic to port 80.

Example. Filter traffic to port 80.

- Click New.
- Name the filter: "tcp destination port 80".
- Remove the word "new" from the filter string field.
- Click the "+ Expression" button to get a list of protocols that can be displayed.
- Scroll down to "TCP" (Transmission Control Protocol) in the field name and click the left arrow to get sub listings of TCP.
- Select tcp.dstport – Destination Port.

Note: There are various types of TCP port field names that can be used. This variety can also be seen with the other protocols, e.g., Internet Protocol (IP), Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP), and Address Resolution Protocol (ARP).

- Click == in Relation field.
- Type "80" into the field under "Value"
- Click "OK"
- Click "Save"
- Click "Apply" to apply the new filter to the data.
- Click "Close" to close the dialog box.

All the data with the TCP destination port of 80 will show.

In order to use a variation of the new filter to view FTP traffic on TCP destination port 21, click inside the filter tool bar text box and change 80 to 21, and then click Apply.

To remove a display filter and view the data unfiltered in Ethereal, in the filter tool bar click Clear⁵.

⁴ Sharpe, Richard. <http://www.ethereal.com/docs/user-guide-sp/#ChUseFilterToolbarSection>

⁵ For more information on using Display filters see:

Sharpe, Richard. <http://www.ethereal.com/docs/user-guide-sp/#ChUseFilterToolbarSection>; and, Orebaugh, Angela. Ethereal Packet Sniffing. Syngress, 2004. Pg. 172

Results of Viewing the Captured Data with Display Filters

For purposes of this test, the destination is the server from which the log files were examined with IP address of 66.92.xx.xx unless otherwise noted.

The port filter used for TCP traffic in this test was `tcp.port == xy` in order to obtain traffic from either the source or destination port. The same was done with UDP traffic `udp.port == xy`.

These are the results of an eight day traffic capture, Feb. 9, 2005, 21:52 hrs through Feb 18, 2005, 10:23 hrs.

Results

4297 packets were captured during this test. 76.03 % of the data was from TCP; 21.6% was from UDP; 1.56% was from ICMP, and 0.81% was from ARP.

Only the two highest traffic protocols of TCP & UDP are listed in this paper. In TCP, the two largest packet captures each have a graph; and in addition, the largest UDP packet capture also has a graph.

Category 1, TCP Port⁶ traffic:

`tcp.port == 21` Protocol FTP as destination port with one exception as the source port, 17 packets total. Possible Ramen Worm⁷.

`tcp.port == 22` SSH destination port, 32 packets, source ports varied, but nearly all were high five digit numbers.

`tcp.port == 23` Telnet no traffic.

`tcp.port == 25` SMTP traffic as destination port, 22 packets, from five source IP's, source ports varied, but nearly all were high five digit numbers.

`tcp.port == 42` Name Server as destination port, six packets from port 80 HTTP, one frame from port 3389 MS Term Serv, the remaining frame from port 6000 Xwindows, total of eight packets of traffic.

`tcp.port == 53` DNS traffic as destination port, 4 packets.

`tcp.port == 79` Finger no traffic.

`tcp.port == 80` HTTP destination port, 21 packets, one IP tried eight times from source port 4293.

`tcp.port == 110` POP3 no traffic.

⁶ Scrambray, Joel. Hacking Exposed: Network Security Secrets & Solutions, 2nd Edition. Berkeley, 2001. Pages 658 -660.

⁷ Orebaugh, Angela. Ethereal Packet Sniffing. Syngress, 2004. Pg. 371.

tcp.port == 111 Sun RPC traffic destination port, 3 consecutive packets.

tcp.port == 135 NT RPC or DCE endpoint resolution destination port, 583 packets, from multiple IP's, with several blocks of same source IP, but different source ports. This port shows up under its own Information heading of "epmap" in Ethereal.

The screenshot displays the Ethereal interface with a filter set to 'tcp.port==135'. The main pane shows a list of 324 packets, all with destination port 135. The packet details pane shows a SYN packet from source 66.92.xx.xx to destination 66.92.xx.xx on port 135. The packet bytes pane shows the raw data of the SYN packet.

No.	Time	Source -	Destination	Protocol	Info
4	177.674462	204.60.xx.xx	66.92.xx.xx	TCP	1974 > epmap [SYN] Seq=0 Ack=0 win=64240 Len=0
37	12884.881403	65.147.xx.xx	66.92.xx.xx	TCP	1825 > epmap [SYN] Seq=0 Ack=0 win=8160 Len=0 M
42	15670.799684	66.91.xx.xx	66.92.xx.xx	TCP	3273 > epmap [SYN] Seq=0 Ack=0 win=64240 Len=0
46	16025.456487	66.0.xx.xx	66.92.xx.xx	TCP	3205 > epmap [SYN] Seq=0 Ack=0 win=64240 Len=0
47	16028.459876	66.0.xx.xx	66.92.xx.xx	TCP	3205 > epmap [SYN] Seq=0 Ack=0 win=64240 Len=0
55	16747.710745	212.160.xx.xx	66.92.xx.xx	TCP	3708 > epmap [SYN] Seq=0 Ack=0 win=64240 Len=0
85	22275.949465	65.148.xx.xx	66.92.xx.xx	TCP	3447 > epmap [SYN] Seq=0 Ack=0 win=8760 Len=0 M
130	30898.158405	66.91.xx.xx	66.92.xx.xx	TCP	3866 > epmap [SYN] Seq=0 Ack=0 win=64240 Len=0
149	33424.741106	66.81.xx.xx	66.92.xx.xx	TCP	4592 > epmap [SYN] Seq=0 Ack=0 win=8760 Len=0 M
150	33427.772479	66.81.xx.xx	66.92.xx.xx	TCP	4592 > epmap [SYN] Seq=0 Ack=0 win=8760 Len=0 M
158	35496.445632	172.182.xx.xx	66.92.xx.xx	TCP	3634 > epmap [SYN] Seq=0 Ack=0 win=32767 Len=0
160	35930.919351	80.136.xx.xx	66.92.xx.xx	TCP	2744 > epmap [SYN] Seq=0 Ack=0 win=16384 Len=0
239	48555.727011	83.145.xx.xx	66.92.xx.xx	TCP	27954 > epmap [SYN] Seq=0 Ack=0 win=65535 Len=0
249	49732.207005	66.90.xx.xx	66.92.xx.xx	TCP	3805 > epmap [SYN] Seq=0 Ack=0 win=64240 Len=0
250	49734.960192	66.90.xx.xx	66.92.xx.xx	TCP	3805 > epmap [SYN] Seq=0 Ack=0 win=64240 Len=0
251	49741.148537	66.90.xx.xx	66.92.xx.xx	TCP	3805 > epmap [SYN] Seq=0 Ack=0 win=64240 Len=0
282	56908.308045	66.92.xx.xx	66.92.xx.xx	TCP	1704 > epmap [SYN] Seq=0 Ack=0 win=65268 Len=0
291	56910.956430	66.92.xx.xx	66.92.xx.xx	TCP	1704 > epmap [SYN] Seq=0 Ack=0 win=65268 Len=0
298	56916.872253	66.92.xx.xx	66.92.xx.xx	TCP	1704 > epmap [SYN] Seq=0 Ack=0 win=65268 Len=0
306	57401.250925	68.239.xx.xx	66.92.xx.xx	TCP	3977 > epmap [SYN] Seq=0 Ack=0 win=16384 Len=0
307	57693.131253	66.69.xx.xx	66.92.xx.xx	TCP	1924 > epmap [SYN] Seq=0 Ack=0 win=16384 Len=0
323	61468.937258	172.201.xx.xx	66.92.xx.xx	TCP	1656 > epmap [SYN] Seq=0 Ack=0 win=64240 Len=0
324	61571.571833	66.167.xx.xx	66.92.xx.xx	TCP	1191 > epmap [SYN] Seq=0 Ack=0 win=16384 Len=0

Time to live: 124
 Protocol: TCP (0x06)
 Header checksum: 0x48c4 (correct)
 Source: 66.92.xx.xx (66.92.xx.xx)
 Destination: 66.92.xx.xx (66.92.xx.xx)
 Transmission Control Protocol, Src Port: 3812 (3812), Dst Port: epmap (135), Seq: 0, Ack: 0, Len: 0
 Source port: 3812 (3812)
 Destination port: epmap (135)

0000 00 40 2b 66 d6 4c 00 90 1a 40 a2 9f 08 00 45 20 .@+F.L...@...E
 0010 00 30 ed f1 40 00 7c 06 48 c4 42 5c f9 bb 42 5c .0.@.].H.B...B\
 0020 49 7e 0e e4 00 87 9a 7e c2 12 00 00 00 00 70 02 I~.....p.
 0030 ff ff 4f 31 00 00 02 04 05 b4 01 01 04 02 ..01.....

Figure 2. TCP 135 NT RPC

tcp.port == 137 NetBIOS⁸ Name Service port, no traffic.

tcp.port == 139 NETBIOS Session Service destination port, 239 packets, many different source IP's, some same source IP's for two or three scans. This shows up under its own information heading of "netbios-ssn" in Ethereal.

tcp.port == 143 Imap no traffic.

tcp.port == 162 SNMP-Trap two packets.

tcp.port == 328 Common Name unassigned⁹, seven packets as destination port with a source port of 80 HTTP, 3 different groupings of source IP's.

tcp.port == 389 Imap two packets.

tcp.port == 443 HTTPS, SSL, 4 packets.

⁸ IANA, <http://www.iana.org/assignments/port-numbers>

⁹ IANA, <http://www.iana.org/assignments/port-numbers>

tcp.port == 445 Microsoft SMB destination port, 1524 packets, varied source IP's. 114 Packets with one source IP that tried a large variety of ports. Another source IP scanned for 103 packets, and another for 91 packets.

No.	Time	Source	Destination	Protocol	Info
2	147.322546	66.178.xx.xx	66.92.xx.xx	TCP	3872 > microsoft-ds [SYN] Seq=0 Ack=0 win=16384
3	150.173873	66.178.xx.xx	66.92.xx.xx	TCP	3872 > microsoft-ds [SYN] Seq=0 Ack=0 win=16384
10	1969.521997	201.138.xx.xx	66.92.xx.xx	TCP	2450 > microsoft-ds [SYN] Seq=0 Ack=0 win=64240
11	1972.334609	201.138.xx.xx	66.92.xx.xx	TCP	2450 > microsoft-ds [SYN] Seq=0 Ack=0 win=64240
12	2147.597524	216.78.xx.xx	66.92.xx.xx	TCP	3085 > microsoft-ds [SYN] Seq=0 Ack=0 win=8160 L
13	2151.086261	216.78.xx.xx	66.92.xx.xx	TCP	3085 > microsoft-ds [SYN] Seq=0 Ack=0 win=8160 L
14	2586.820583	84.121.xx.xx	66.92.xx.xx	TCP	3904 > microsoft-ds [SYN] Seq=0 Ack=0 win=16384
21	4808.639200	66.116.xx.xx	66.92.xx.xx	TCP	4945 > microsoft-ds [SYN] Seq=0 Ack=0 win=16384
22	4811.521839	66.116.xx.xx	66.92.xx.xx	TCP	4945 > microsoft-ds [SYN] Seq=0 Ack=0 win=16384
23	4884.090563	85.84.xx.xx	66.92.xx.xx	TCP	3125 > microsoft-ds [SYN] Seq=0 Ack=0 win=64240
24	4886.955019	85.84.xx.xx	66.92.xx.xx	TCP	3125 > microsoft-ds [SYN] Seq=0 Ack=0 win=64240
25	5527.581537	61.231.xx.xx	66.92.xx.xx	TCP	3301 > microsoft-ds [SYN] Seq=0 Ack=0 win=64800
26	5530.552683	61.231.xx.xx	66.92.xx.xx	TCP	3301 > microsoft-ds [SYN] Seq=0 Ack=0 win=64800
27	5536.565261	61.231.xx.xx	66.92.xx.xx	TCP	3301 > microsoft-ds [SYN] Seq=0 Ack=0 win=64800
30	10880.528358	66.245.xx.xx	66.92.xx.xx	TCP	1701 > microsoft-ds [SYN] Seq=0 Ack=0 win=64240
31	10883.455188	66.245.xx.xx	66.92.xx.xx	TCP	1701 > microsoft-ds [SYN] Seq=0 Ack=0 win=64240
39	14571.480328	210.165.xx.xx	66.92.xx.xx	TCP	63752 > microsoft-ds [SYN] Seq=0 Ack=0 win=64240
40	14579.140151	210.165.xx.xx	66.92.xx.xx	TCP	61640 > microsoft-ds [SYN] Seq=0 Ack=0 win=64240
45	15981.338911	24.28.xx.xx	66.92.xx.xx	TCP	4079 > microsoft-ds [SYN] Seq=0 Ack=0 win=16384
52	16361.561980	66.38.xx.xx	66.92.xx.xx	TCP	4696 > microsoft-ds [SYN] Seq=0 Ack=0 win=16384
53	16364.461762	66.38.xx.xx	66.92.xx.xx	TCP	4696 > microsoft-ds [SYN] Seq=0 Ack=0 win=16384
57	16871.884412	141.150.xx.xx	66.92.xx.xx	TCP	2720 > microsoft-ds [SYN] Seq=0 Ack=0 win=16384
58	16874.300834	141.150.xx.xx	66.92.xx.xx	TCP	2720 > microsoft-ds [SYN] Seq=0 Ack=0 win=16384
59	16880.909603	141.150.xx.xx	66.92.xx.xx	TCP	2720 > microsoft-ds [SYN] Seq=0 Ack=0 win=16384
64	17337.069103	66.120.xx.xx	66.92.xx.xx	TCP	2727 > microsoft-ds [SYN] Seq=0 Ack=0 win=64240

▶ Frame 2 (62 bytes on wire (62 bytes captured) on interface eth0)

▶ Ethernet II, Src: 00:90:1a:40:a2:9f, Dst: 00:40:2b:66:d6:4c

▶ Internet Protocol, Src Addr: 66.178.xx.xx (66.178.xx.xx), Dst Addr: 66.92.xx.xx (66.92.xx.xx)

▶ Transmission Control Protocol, Src Port: 3872 (3872), Dst Port: microsoft- (445), Seq: 0, Len: 0

```

0000  00 40 2b 66 d6 4c 00 90 1a 40 a2 9f 08 00 45 20  .@+f.L..@....E
0010  00 30 f2 25 40 00 76 06 e8 d3 42 b2 5b 22 42 5c  .0.%@.v..B.["B\
0020  49 7e 0f 20 01 bd 85 69 d3 63 00 00 00 70 02  I~. .i.c....p.
0030  40 00 af c6 00 00 02 04 05 b4 01 01 04 02  @.....
  
```

Figure 3. TCP 445 Microsoft SMB

445 Microsoft SMB destination port shows up under its own Information heading of "Microsoft-ds" in Ethereal.

tcp.port == 515 printer destination port, 3 consecutive packets, possible Ramen¹⁰.

tcp.port == 554 Real Time Stream Control Protocol.

Destination port, 23 packets, from 12 different source IP's.

tcp.port == 943 unassigned destination port, 17 packets, 5 incremental source IP's, all source ports are port 80 HTTP.

tcp.port == 1025 network blackjack destination port, 21 packets, 8 source IP's.

tcp.port == 1080 Socks Socks dozen packets.

tcp.port == 1313 bmc-patrol Source port, destination port 445 MS SMB, 1 frame.

¹⁰ Orebaugh, Angela. Ethereal Packet Sniffing. Syngress, 2004. Pg. 371.

tcp.port == 1243 SubSeven port, destination or source port, 4 packets, from two different IP's. Two packets with source port of 1243 went to destination port of 445 MS NetBIOS. Other two packets had source port of 4132 NUTS Daemon is default and destination port of 1243.

tcp.port == 1433 MS SQL destination port, 96 packets, varied IP's, one block of traffic had same source IP for 9 packets, using 5 different source ports. This shows up under its own Information heading of "ms-sql-s" in Ethereal.

tcp.port == 1434 Microsoft SQL Monitor no traffic.

tcp.port == 2301 Compaq-web three packets source port, destination port 445 MS SMB.

tcp.port == 3300 bmc-patrol-agent two frame source port, destination port 445 MS SMB.

tcp.port == 2745 Bagel back door¹¹, destination port, nine packets.

tcp.port == 3127 My Doom back door¹², 12 packets, from five IP addresses, two had source ports of 3127, and the remaining had destination ports of 3127. The two packets with source port of 3127 had a destination port of 445 MS SMB destination.

tcp.port == 3198 My Doom back door¹³, no traffic.

tcp.port == 3306 MySQL destination port 37 packets, 21 source IP's, SQL worm. MySQL UDF Worm¹⁴.

tcp.port == 3351 ssql source port, five packets, two groupings, consecutive packets, destination port 445.

tcp.port == 3389 MS Term Serv, both source and destination port.

Destination port, nine packets, 5 source IP's.

Source port, six packets, 3 source IP's.

tcp.port == 4001 Cisco-Mgmt source port, two consecutive frames, destination port 445 MS SMB.

tcp.port == 4045 NFS-Lockd two consecutive packets source port, destination port 135 NT RPC.

tcp.port == 4899 Radmin remote access port¹⁵, destination port, 109 packets, 12 packets from same IP address.

¹¹ Nazario, Jose. http://ims.eecs.umich.edu/worm_report/

¹² Nazario, Jose. http://ims.eecs.umich.edu/worm_report/

¹³ Nazario, Jose. http://ims.eecs.umich.edu/worm_report/

¹⁴ US-Cert, http://www.us-cert.gov/current/current_activity.html#MySQLUDF

¹⁵ Scheidell, Michael. <http://www.securityfocus.com/archive/1/290099/2002-09-01/2002-09-07/0>

tcp.port == 5631 PCAnywhere no traffic.
tcp.port == 5800 VNC no traffic.
tcp.port == 5900 RealVNC no traffic.

tcp.port == 6000 XWindows, source port for 12 packets of traffic; six had destination port of 1433 MS SQL; two had destination of port 4899 Radmin, one a destination port of 42 Name Server; one a destination port of 3389 MS Term Serv; one a destination port of 6129.

tcp.port == 6667 IRC clients, common Trojan port, 7 packets, all from same IP address. Variety of Exploits:

- W32.gaobot.cii
- Backdoor.lateda.b
- Protoride.b
- Backdoor.Alcani
- W32.spybot.dnb
- W32.Randex.ATS
- W32.korgo.a
- SubSeven¹⁶

tcp.port == 6711 - 6713 SubSeven¹⁷ ports, no traffic.
tcp.port == 6776 SubSeven port¹⁸, no traffic.
tcp.port == 6777 Bagle.A port¹⁹, no traffic.

tcp.port == 7000 Exploit port variety of attacks. Traffic 53 packets source port and 29 packets destination port.

- Exploit Translation Server, Kazimas,
- Remote Grab, SubSeven,
- SubSeven 2.1 Gold²⁰

tcp.port == 8000 Web applications, destination port, 5 packets.
tcp.port == 8080 Web applications, 7 packets (5 from same IP address), and destination port.

tcp.port == 8420 Unassigned²¹ port as a common name, destination port for all 13 packets, 8 different source IP's (sometimes the same source IP will try the same attack several times).

tcp.port == 12345 NetBus²² No traffic.

¹⁶ <http://www.doshelp.com/Ports/6667.htm>

¹⁷ SANS Institute. [Track 1 - Internet Security Technologies](#) Jan 2004. Pg 166.

¹⁸ SANS Institute. [Track 1 - Internet Security Technologies](#) Jan 2004. Pg 166.

¹⁹ Nazario, Jose. http://ims.eecs.umich.edu/worm_report/

²⁰ <http://www.blackcode.com/trojans/ports.php?port=7000>

²¹ IANA, <http://www.iana.org/assignments/port-numbers>

²² SANS Institute. [Track 1 - Internet Security Technologies](#) Jan 2004. Pg. 162.

tcp.port == 12346 NetBus²³ No traffic.
tcp.port == 27374 default SubSeven²⁴ & Ramen²⁵ port no traffic.
tcp.port == 31337 Back Orifice²⁶ no traffic.
tcp.port == 32771 rcp-solaris no traffic.
tcp.port == 43188 reachout no traffic.

tcp.port == 50736 Unknown²⁷ common port, destination port for all 17 packets (10 different source IP's, one IP tried same attack six consecutive times, some IP's repeated non-consecutive traffic. Eleven packets from port 80 HTTP, three packets from port 7000, one from port 7777, one from port 21 (FTP), one from port 4000.

tcp.port == 65301 PCAnywhere – def no traffic
tcp.port == 65535 or 0, Sons of Jackal²⁸ no traffic

Category 2, UDP port traffic

udp.port == 53 DNS²⁹ no traffic
udp.port == 69 TFTP³⁰ no traffic
udp.port == 135 DCE endpoint resolution no traffic

udp.port == 137 NetBIOS Name Server³¹ destination port, 577 packets, largest block of same source IP is 20 packets, the packets were only consecutive a few at a time. It appears this Source IP retried its attack every thousand packets or so.

The scan has the characteristics of older port scanning worm network.vbs³² and its derivatives.

²³ SANS Institute. [Track 1 - Internet Security Technologies](#) Jan 2004. Pg. 162.

²⁴ SANS Institute. [Track 1 - Internet Security Technologies](#) Jan 2004. Pg. 163

²⁵ Orebaugh, Angela. [Ethereal Packet Sniffing](#). Syngress, 2004. P. 371

²⁶ SANS Institute. [Track 1 - Internet Security Technologies](#) Jan 2004. Pg. 167.

²⁷ IANA, <http://www.iana.org/assignments/port-numbers>

²⁸ SANS Institute. [Track 1 - Internet Security Technologies](#) Jan 2004. Pg. 173.

²⁹ IANA, <http://www.iana.org/assignments/port-numbers>

³⁰ IANA, <http://www.iana.org/assignments/port-numbers>

³¹ IANA, <http://www.iana.org/assignments/port-numbers>

³² Alexander, Bryce. http://www.sans.org/resources/idfaq/port_137.php

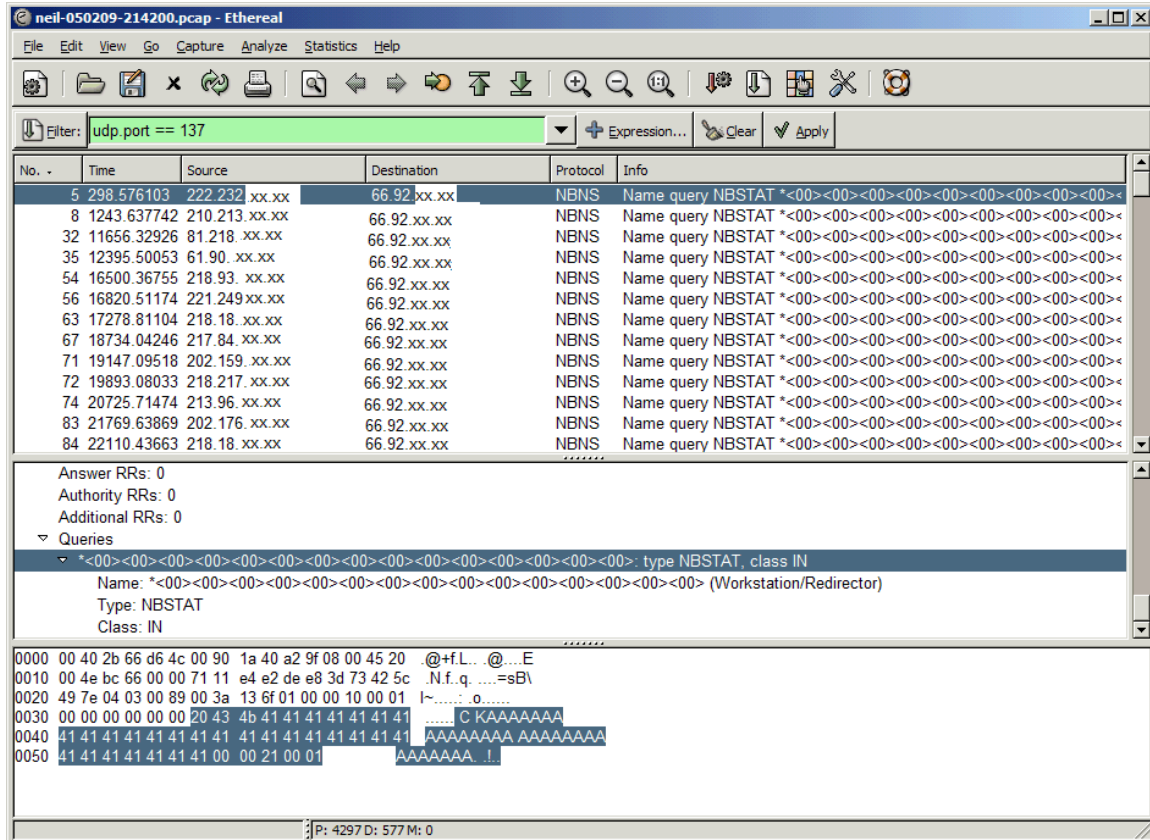


Figure 4. UDP 137 NetBIOS Name Server

The Ethereal Summary window Information column per packet contains:

Name query NBSTAT

*<00><00><00><00><00><00><00><00><00><00><00><00><00><00><00><00>

The below, is from the Ethereal Data View Window of packet 5; however, the characteristic “CKAAAAAAAA” repeats itself in each packet.

```
0000 00 40 2b 66 d6 4c 00 90 1a 40 a2 9f 08 00 45 20 .@+f.L...@....E
0010 00 4e e5 f5 00 00 76 11 49 75 53 2b 36 0f 42 5c .N....v.luS+6.B\
0020 49 7e 4c c3 00 89 00 3a 5d ff 00 d1 00 10 00 01 I~L.....:];.....
0030 00 00 00 00 00 00 20 43 4b 41 41 41 41 41 41 41 ..... CKAAAAAAAA
0040 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAAAAAA
0050 41 41 41 41 41 41 41 00 00 21 00 01 AAAAAAA.!..
```

Verified NetBIOS traffic is only to destination port by using “udp.dstport” in Ethereal display filter as well. This traffic shows up under its own Protocol heading of NBNS in Ethereal.

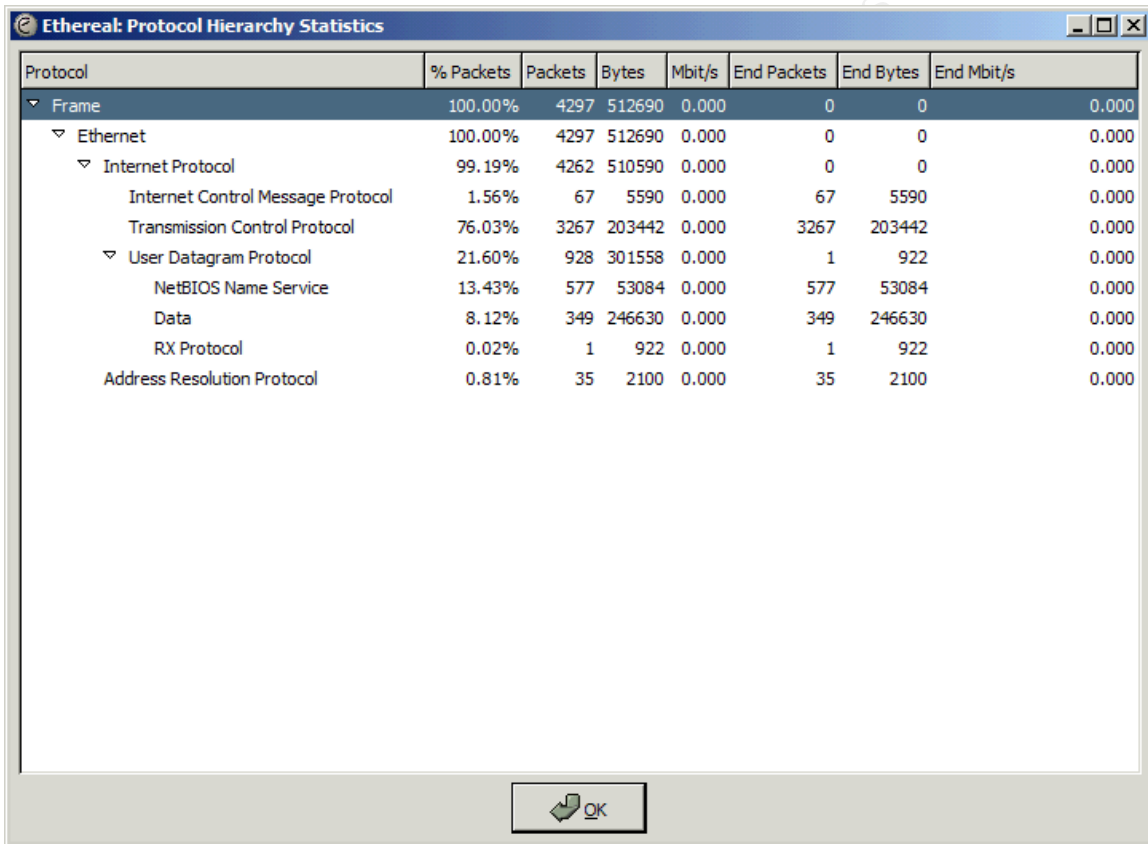
udp.port == 139 NetBIOS Session Service³³ no traffic.

³³ IANA, <http://www.iana.org/assignments/port-numbers>

udp.port == 445 Microsoft SMB³⁴ no traffic.
 udp.port == 1433 Microsoft-SQL-Server³⁵ no traffic.
 udp.port == 1434 Microsoft SQL Monitor³⁶, destination port, 69 packets.

udp.port == 31337 Backorifice³⁷ no traffic.
 udp.port == 27374 default SubSeven³⁸ no traffic.

A summary of all the protocols found by Ethereal by the menu bar, Statistics - Protocol Hierarchy graph:



Protocol	% Packets	Packets	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
Frame	100.00%	4297	512690	0.000	0	0	0.000
Ethernet	100.00%	4297	512690	0.000	0	0	0.000
Internet Protocol	99.19%	4262	510590	0.000	0	0	0.000
Internet Control Message Protocol	1.56%	67	5590	0.000	67	5590	0.000
Transmission Control Protocol	76.03%	3267	203442	0.000	3267	203442	0.000
User Datagram Protocol	21.60%	928	301558	0.000	1	922	0.000
NetBIOS Name Service	13.43%	577	53084	0.000	577	53084	0.000
Data	8.12%	349	246630	0.000	349	246630	0.000
RX Protocol	0.02%	1	922	0.000	1	922	0.000
Address Resolution Protocol	0.81%	35	2100	0.000	35	2100	0.000

Figure 5. Protocol Hierarchy Summary Graph

³⁴ IANA, <http://www.iana.org/assignments/port-numbers>

³⁵ IANA, <http://www.iana.org/assignments/port-numbers>

³⁶ IANA, <http://www.iana.org/assignments/port-numbers>

³⁷ SANS Institute. [Track 1 - Internet Security Technologies](#) Jan 2004. Pg. 167.

³⁸ SANS Institute. [Track 1 - Internet Security Technologies](#) Jan 2004. Pg. 166.

Conclusion

Traffic usually repeated itself from the same IP and same source port two to three consecutive times against a specific destination port. For example, source IP 218.30.21.xxx, TCP source port http, destination port 328 showed up two times in a row twice: frames 1177 & 1178; and 1182 & 1183. One possible explanation of this may be that the attacker hopes to protect against dropped or timed-out traffic per connection. For example: the attacker attempts a connection, SYN, several times using the same method and hopes that one attempt should be successful with an SYN/ACK.

Many of the attacks were against recent well published vulnerabilities. These were mostly Microsoft with some Linux as well. However, any system is vulnerable to attack.

In Ethereal display filter, the query `tcp.flags.syn==1&&tcp.flags.ack==1`³⁹ showed three or more attempted connections against open ports:

7000
50736
8420
328
943

One interpretation of this attack style is:

The attackers seem to have tricks to get through firewalls, including sending various types of malformed packets. A firewall may be configured not to let new connections in, so an attacker will set the "ACK" flag to try making any such firewalls along the way think that the packet is part of an open TCP connection. The packets erroneously show up in your analysis as belonging to open connections for this same reason⁴⁰.

However, in this case, the IP did not send out a single packet. "One cannot have a working TCP connection without data flowing in both directions."⁴¹

"The other thing that some naively-configured firewalls do is accept traffic from well-known ports. For example, your firewall may accept traffic from port 3389 so that you can successfully connect to somebody else's Microsoft Terminal Server⁴²." This was partially seen in the TCP traffic captured and noted earlier going to port 3389. "Because the attackers have

³⁹ Orebaugh, Angela. Ethereal Packet Sniffing. Syngress, 2004. Pg. 355.

⁴⁰ Madden, Patrick. E-mail. March 2005.

⁴¹ Madden, Patrick. E-mail. March 2005.

⁴² Madden, Patrick. E-mail. March 2005.

complete control over source port, they try using source port numbers that a naively-configured firewall will allow through, thinking that it's a connection the user opened.⁴³ The main way to counter the above threat would be to place the server behind a firewall on a private network (10.x.x.x for example), and to have the trusted users or machines connect to the servers via a gatekeeper service, such as a VPN (Virtual Private Network) or a TCP connection tunneled through SSH.

I also expected to see incremental increasing source IP addresses due to IP spoofing. This was not the case.

One of the disappointments of Ethereal was that I could not find a simple way to get an automated count of the exact number of packets used by a specific protocol and port, e.g., TCP port 7000. I am not familiar with Tcpdump or WinDump, and was seeking a way to get a text file from Ethereal that could be imported into an application like MS Excel and be manipulated. It took some time to play with all text exporting options to find what I was seeking. I include the following to save a person the same effort in the future.

While reading Display file in Ethereal

- click File
- Export as "Plain Text file",
- uncheck Packet Details,
- type in path and name of file,
- click OK.

When open this text file in Excel,

- choose Delimited,
- with a delimiter of a space.

⁴³ Madden, Patrick. E-mail. March 2005.

References

1. Alexander, Bryce. "Intrusion Detection FAQ: Port 137 Scan." SANS. May 10, 2000. URL: http://www.sans.org/resources/idfaq/port_137.php
2. "Show Trojan By Port Number." Blackcode Trojan Library. March 12, 2005. URL: <http://www.blackcode.com/trojans/ports.php>
3. "Known Trojans/Worms for Port: 6667." Doshelp.com Known Trojans/Worms for Port: 6667. Feb 2005. URL: <http://www.doshelp.com/Ports/6667.htm>
4. Hazeleger, Dick. "Packet Sniffing: A 'Crash course'." 2003. URL: <http://security.hazeleger.net/>
5. IANA, "Port Numbers." February 23, 2005. URL: <http://www.iana.org/assignments/port-numbers>
6. Nazario, Jose, and Evan Cooke. "Increased Bagle and MyDoom Backdoor Scanning Activity." Arbor Networks. March 26, 2004. URL: http://ims.eecs.umich.edu/worm_report/
7. Madden, Patrick. "Re: Preliminary results very rough." E-mail reply to Neil Orlando from computer security consultant giving feed back on findings after data analysis by author for this practical paper. March 7, 2005. 5:38pm.
8. OPENXTRA Limited. "Ethereal - Setting a Display Filter." URL: <http://www.openxtra.co.uk/support/howto/ethereal-display-filter.php>
9. OPENXTRA Limited. "Getting Started with Ethereal." May 2003. URL: <http://www.openxtra.co.uk/support/documentation/ethereal-getting-started-guide.pdf>
10. Orebaugh, Angela, Greg Morris, Ed Warnicke, and Gilbert Ramirez. Ethereal Packet Sniffing. Rockland: Syngress, 2004.
11. SANS Institute. Track 1 - Internet Security Technologies, Volume 1.3. SANS Press , Jan 2004.
12. Scheidell, Michael. "SECNAP Security Alert: Radmin Default install options vulnerability." SecurityFocus. September 2, 2002. URL: <http://www.securityfocus.com/archive/1/290099/2002-09-01/2002-09->

[07/0.](#)

13. Scambray, Joel, Stuart McClure, George Kurtz. Hacking Exposed: Network Security Secrets & Solutions, 2nd Edition. Berkeley: Osbourne/McGraw-Hill, 2001.
14. Sharpe, Richard, Ed Warnicke, Ulf Lamping. "Ethereal User's Guide." 2004. URL:
<http://www.ethereal.com/docs/user-guide/>
15. US-CERT, "US-Cert Current Activity." March 11, 2005 15:05:14 EST. URL:
http://www.us-cert.gov/current/current_activity.html

© SANS Institute 2000 - 2005, Author retains full rights.



Upcoming SANS Training

[Click Here for a full list of all Upcoming SANS Events by Location](#)

SANS Cyber Defence Canberra 2017	Canberra, AU	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MDUS	Jun 26, 2017 - Jul 01, 2017	Live Event
SEC564:Red Team Ops	San Diego, CAUS	Jun 29, 2017 - Jun 30, 2017	Live Event
SANS London July 2017	London, GB	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, JP	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS ICS & Energy-Houston 2017	Houston, TXUS	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, SG	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Los Angeles - Long Beach 2017	Long Beach, CAUS	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Munich Summer 2017	Munich, DE	Jul 10, 2017 - Jul 15, 2017	Live Event
SANSFIRE 2017	Washington, DCUS	Jul 22, 2017 - Jul 29, 2017	Live Event
Security Awareness Summit & Training 2017	Nashville, TNUS	Jul 31, 2017 - Aug 09, 2017	Live Event
SANS San Antonio 2017	San Antonio, TXUS	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Hyderabad 2017	Hyderabad, IN	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MAUS	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Prague 2017	Prague, CZ	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UTUS	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS New York City 2017	New York City, NYUS	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Chicago 2017	Chicago, ILUS	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Adelaide 2017	Adelaide, AU	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VAUS	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS San Francisco Fall 2017	San Francisco, CAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FLUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NVUS	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, IE	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS Paris 2017	OnlineFR	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced