Global Information Assurance Certification Paper

Free Tools for Network Security
Jeffrey Shuron
May 16, 2001
Version 1.2d

Congratulations! You've just been appointed the person in charge of information and network security for your company. Not only do you need to ascertain what is running on your network from a hardware and software standpoint, but also what security risks exist and how to eliminate them. The tools that allow someone to accomplish the task of finding security breaches in a network are numerous indeed. While a good number of these tools have a price tag associated with them, there are a number of free tools that you can download to aid in the never-ending struggle to locate and patch security vulnerabilities. We will take a look at Nmap (the popular port scanner), and Ethereal (a packet sniffer). Both are excellent tools that will prove advantageous many times over.

One caveat that must be mentioned is what operating systems these tools will run on. Nmap will only run on Unix, or a Unix-type platform. Ethereal will run on either a Windows or a Unix-type system. To install either on a Unix or Unix-type platform please consult the installation documentation at the specific web site.

NMAP

Nmap is the popular port scanner courtesy of Fyodor at www.insecure.org. According to a Security Watch article in InfoWorld from July of 1998, "If your goal is to understand your network from a 40,000-foot view, then Windows port scanning tools will suffice. But if you're serious about your security and looking for the holes that crackers will find, then take the time to install a Linux box and use nmap."[1]

So what does Nmap do, and why is it so useful? Nmap is a utility that will allow an individual to scan an entire network or a single host, allowing you to discover what services are running on individual boxes along with what hosts are actually up and running. This tool will give you the opened ports on a particular host, allowing you to see what service someone could connect to. Important? You bet it is! Let's say that a new server farm is about to go live on your network. Are you going to allow someone to place a new server on your network without knowing what ports it's listening on? Let's hope not! Nmap allows you to perform various types of scans to gather information about your network so that you can take the appropriate measure to make sure that the machines are as secure as possible.

One note before we continue. We're not going to go in depth on TCP, UDP, and ICMP protocols. If any part of this paper raises question, please consult your favorite tome.

Before we begin to run some sample scans, we'll take a look into the types of scans Nmap can perform, and what some of the command line options are (There is a GUI interface for Nmap for those who are so inclined[2,3]).

Nmap supports, among others, the following techniques: TCP connect (), TCP SYN (half open), ftp proxy (bounce attack), ICMP (ping sweep), FIN, ACK sweep, Xmas Tree, SYN sweep, UDP, and Null scans. You can also do OS detection via TCP/IP fingerprinting[4], decoy scanning, and fragmentation scanning.

The command line syntax for Nmap looks like this:

Nmap [Scan Type(s)] [Options] <network or host #1…..[#N]>

Here are some of the scan types, followed by a brief description.

-sT TCP connect () scan: The basic TCP scan. It is easily detected, and will generate connection errors for any service which accepts the connection then has it shut down.

-sS TCP SYN scan: A "half-open" scan, as you never open a full TCP connection via the three-way handshake. A SYN is sent, and a SYN | ACK reply means the port port is listening while a RST replay indicates a closed port. A SYN | ACK reply generates a RST to tear down the connection. This type of scan is harder to detect.

-sF Stealth FIN, Xmas Tree, and Null scans: These scans are useful in testing the
-sX ability to scan through a firewall/filtering device undetected. These scans are
-sN recommended for experts.

-sP Ping scanning: This option allows you to perform a ping sweep without doing any actual scanning.

-sU UDP scans: Nmap sends a 0 byte UDP packet to each port on the scanned machine. An ICMP unreachable reply means the port is closed.

-sA ACK scan: This scan is used primarily to map out firewall rule sets, determining if it is doing plain packet filtering, or stateful inspect

-sR RPC scan: Takes all the UDP/TCP ports found and floods them with SunRPC Null command to figure out if they are RPC ports.

Now we will list some general options available with a brief description. While there is no need to add any of the options, you will find some of them valuable when performing scans.

-P0 Don't ping hosts before scanning them. Handy when ICMP echo is disabled through a firewall.

-PT Determine what hosts are up by using TCP ping. Instead of ICMP requests, ACK packets are sent out on the network. A RST response means the host is alive

-PI Uses a true ping packet. Also looks for subnet-directed broadcast addresses on your network.

-O Used to identify the OS of the host via TCP/IP fingerprinting.

-v Verbose mode. This shows you more information about what is taking place during the scan.

-h This option will display a help screen in quick reference format, as the man page for Nmap is quite extensive!

-oN <logfilename>
Logs the results of the scan in readable format to a file you specify.

-iL <inputfilename>
Allows you to read the target(s) specified from a file rather than the command line. The file should contain hosts separated by commas, or spaces, or new lines.

-p Use this option to specify what ports, or range of ports, you want to scan.

-F Use this option to scan only the ports listed in the services file of Nmap. This will give you a much faster scan than trying all 65535 ports per host.

-D <decoy1, decoy2,…..me>
This allows you to make your scan appear to come from multiple hosts at once. Make sure the decoys you are using are up or you could accidentally SYN flood your targets!

-n NEVER do reverse DNS resolution on active IP addresses found.

-R ALWAYS do reverse DNS resolution.

We're not done yet! Now we'll take a look at some timing options that can be used when the default is insufficient.

-T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane>
These are timing options to let Nmap know how you would like to proceed with the scan.
Paranoid mode is extremely slow, and waits 5 minutes between packets. This is ideal for avoidance of IDS systems.
Sneaky is similar, but it only waits 15 seconds between packets.
Polite is used to prevent machines from crashing on the network, waiting 0.4 seconds between packets.
Normal is the Nmap default, trying to run as quickly as possible without missing hosts.
Aggressive mode has a 5-minute waiting period between hosts. It waits no more than 1.25 seconds for a response.
Insane should only be used if you don't mind losing information, or are dealing with a very fast network.

Please note that using the Paranoid or Sneaky options should really be used to test the validity of your own network monitoring tools, or to examine suspicious hosts on the internal network only.

But wait, there's more! We need to cover one more thing before we start having some fun with Nmap; target specification. Target specification covers everything that isn't an argument or option. Basically it's the hosts, or networks that you would like to scan.

Let's say that you want to scan an entire class B network. You could use '192.168.*.*' (remember that in Unix an '*' is interpreted by command shells, so if you use them don't forget to put single quotes around your argument). Another way would be 192.168.0.0/16. You could even specify a range like 192.168.1-5.50-175.

The moment of truth. Now we will take a look at some sample scans and what the command syntax looks like for each, along with parts of the results.

Nmap (V. nmap) scan initiated 2.53 as: nmap -sS -sR -v -D 216.32.74.53,64.58.76.176,216.239.37.100 -oN /mnt/hda1/temp/nmap_3 63.50.x.x
Interesting ports on pool-63.50.x.x.bob.grid.net (63.50.x.x):
(The 1512 ports scanned but not shown below are in state: closed)

Port	State	Service (RPC)
21/tcp	open	ftp
22/tcp	open	ssh
23/tcp	open	telnet
80/tcp	open	http
98/tcp	open	linuxconf
111/tcp	open	sunrpc (rpcbind V2)
113/tcp	open	auth
443/tcp	open	https
513/tcp	open	login
514/tcp	open	shell
6000/tcp	open	X11

As you can see, this host was scanned using the TPC SYN scan (-sS) combined with the RPC scan (-sR). Three decoy IP addresses (-D) were used to evade detection. The file was sent, in readable form (-oN) to the specified file. Our target was IP address 63.50.x.x. Take a look at the ports Nmap returned as "listening". This host could be a combination web/ftp server running some flavor of Linux. If we were "black hats" this information would narrow the scope of attacks that we could run against the target system, such as a web site hack or ftp exploit. Refer to Appendix A for a printout of the netstat command. This shows what the scanned host saw.

Nmap (V. nmap) scan initiated 2.53 as: nmap -sT -oN /mnt/hda1/temp/nmap_1 192.168.20.52
Interesting ports on (192.168.20.52):
(The 1511 ports scanned but not shown below are in state: closed)

Port	State	Service
7/tcp	open	echo
9/tcp	open	discard
13/tcp	open	daytime
17/tcp	open	qotd
19/tcp	open	chargen
25/tcp	open	smtp
80/tcp	open	http
135/tcp	open	loc-srv
139/tcp	open	netbios-ssn
443/tcp	open	https
1025/tcp	open	listen

1433/tcp open ms-sql-s

This test host was scanned using the basic TCP connect () scan (-sT), and the output was sent to a specific file for late dissection (-oN). The one opened port that stands out is

TCP port 1433. This indicates that this host is running a version of SQL Server from Microsoft. One possible exploit could be an attempt to connect to the server and logon using the default administrator username "sa" and a blank password.

Great, you say. So how can I apply this to my network? Glad you asked! In every company there is always some individual(s) that load extra services on their machines for whatever reason. One part of any defense-in-depth strategy is to make sure all hosts are running **only** the services necessary. So let's take this one step further and look at 3 scans of a network (one sample each from the last two). The first scan is a ping sweep to see what hosts are alive on the network. The second scan uses the OS fingerprinting scan, and a TCP scan without pinging the hosts. The third scan is a UDP scan. I'll explain the method to my madness at the conclusion of the examples.

Nmap (V. nmap) scan initiated 2.53 as: nmap -sP -v -oN /mnt/hda1/temp/nmap_ping 192.168.x.0/24
Host (192.168.x.0) seems to be a subnet broadcast address (returned 6 extra pings). Still scanning it due to positive ping response from its own IP.
Host (192.168.x.1) appears to be up.
Host (192.168.x.2) appears to be up.
Host (192.168.x.3) appears to be up.
Host (192.168.x.5) appears to be up.
Host (192.168.x.6) appears to be up.
Host (192.168.x.7) appears to be up.
Host (192.168.x.8) appears to be up.
Host (192.168.x.9) appears to be up.
Host (192.168.x.15) appears to be up.
Host (192.168.x.30) appears to be up.
Host (192.168.x.31) appears to be up.
Host (192.168.x.50) appears to be up.
Host (192.168.x.51) appears to be up.
Host (192.168.x.54) appears to be up.
Host (192.168.x.55) appears to be up.
Host (192.168.x.56) appears to be up.
Host (192.168.x.200) appears to be up.

Interesting ports on (192.168.20.54):
(The 1518 ports scanned but not shown below are in state: closed)
Port State Service
135/tcp open loc-srv
139/tcp open netbios-ssn
445/tcp open microsoft-ds
1025/tcp open listen
1058/tcp open nim

TCP Sequence Prediction: Class=random positive increments

Difficulty=11216 (Worthy challenge)

Sequence numbers: 945F7F34 94607639 946155F8 94625263 9462ECCD 94638607
Remote operating system guess: Windows 2000 RC1 through final release

Interesting ports on (192.168.x.3):
(The 1442 ports scanned but not shown below are in state: closed)

Port	State	Service
137/udp	open	netbios-ns
138/udp	open	netbios-dgm
445/udp	open	microsoft-ds
500/udp	open	isakmp
3456/udp	open	vat
5632/udp	open	pcanywherestat

Ok. We see that this is a small network by the number of hosts that responded during the ping sweep. One of the hosts is found to be running Windows 2000. The last scan shows pcAnywhere running on the host. Now, after finding out what each host does from the ping sweep, we find that 192.168.x.3 is a server. A server running pcAnywhere??? If improperly configured, that server could be exploited from another host. My point is that never take anything for granted. You would need to make sure that there is a valid reason for pcAnywhere to be running on a server. If so, take the necessary steps to secure it (passwords, encryption, etc.).

SUMMARY

Nmap is a valuable port scanning tool that can be used to discover the services running on your network hosts. As an integral part of your security toolbox Nmap should be run on a regular basis to uncover rogue applications on unauthorized hosts. Nmap can also be used to ferret out Trojan horses such as SubSeven and Back Orifice. Used properly and wisely, Nmap could be an outstanding addition to your defense-in-depth strategy. If you wish to examine the Namp command line options in depth, please visit www.insecure.org/namp.

One final note. Please make sure you have written permission to run Namp on your network. There is always the possibility that a scan could cause disruption to normal network traffic or even disable machines.

ETHEREAL

Ethereal is a protocol analyzer originally authored by Gerald Combs. It is available from www.ethereal.com. As stated earlier it can be run on either a Windows or Unix-type platform. Ethereal allows you to view real-time packet data from a network by decoding the traffic, or load a file that had been saved from a previous capture. You can also read the capture files of other programs such as LanAlyzer, Sniffer, Sniffer Pro, and NetXray. Ethereal runs as a GUI, but you can run it from a command line using the program Tethereal, which is part of the Ethereal download.

In order to understand Ethereal that much better, a working knowledge of TCP/IP and "sniffing" is key. There is an excellent FAQ[5] by Robert Graham, who was one of the

authors for Network General Corporation's Sniffer Network Analyzer, discussing such things as how to discover if someone is sniffing your network and protocol analysis. An excellent book for reference or learning TCP/IP is Internetworking with TCP/IP Vol. I: Principles, Protocols, and Architecture by Douglas Comer.

First we will take a look at some of the command line options for Ethereal, followed by some examples of capture and display filters, and end with some real-life examples.

Ethereal comes with, among others, the following options:

-c Sets the default number of packets to read when capturing live data.

-f Sets the capture filter expression.

-n Disables network name resolution, such as hostname and TCP or UDP port names.

-r Reads the packet data from file <filename>.

-R If reading a captured file (with the –r option) this option causes the specified display filter to be applied. Packets not matching the filter are discarded.

-Q Has Ethereal exit when a capture session has ended, and is useful in batch mode if using the –c option. This option does require the –i and –w parameters.

-i The name of the network interface or pipe that will be used for capturing live packets. Pipe names should be a named pipe or "-". Libpcap format must be used for data read from pipes.

-w Sets the default capture file name.

-p Takes the interface out of promiscuous mode.

Capture filter syntax and display filter syntax are different, and cannot be substituted for one another. Both allow you to filter out "noise" either during or after a session so that you are able to view only the information that you deem necessary.

Capture filters contain one or more "primitives" and their keywords. A primitive consists of an id, which is a name or number, preceded by one or more qualifiers. The three types of qualifiers are type, dir, and proto. Let's take a closer look:

type What the id or name refers to. **Host**, **port**, and **net** are the possibilities.

dir The transfer direction to and/or from the id. The options are **src**, **dst**, **src or dst**, and **src and dst.** If no dir qualifier exists **src or dst** is the default.

proto Restricts the match to a specific protocol. Possible choices are **ether**, **fddi**, **ip**, **arp**, **rarp**, **decent**, **lat**, **sca**, **moprc**, **mopdl**, **tcp**, and **udp**.

There are special primitives that are the exception. They are **gateway**, **broadcast**, **less**, and **greater**. You create complex filters by using **and**, **not**, and **or** in combination with multiple primitives. Sounds confusing you say? Why don't we take a look at some examples.

src host 192.168.1.100- Will capture packets that come from the host with IP address 192.168.1.100 only.

src net 192.168.1.0/24- Will capture packets if the source IP address has a network number of 192.168.1.x

'gateway test and (port ftp and smtp)'- Will capture all traffic going through the gateway named test that is either on port 21 (ftp) or 25 (smtp). Remember quotes are needed in Unix so the shell won't misinterpret the ( ).

'tcp[13] & 3 != 0 and not src and dst net localnet'- Will show all the start and end packets of every TCP conversation that includes a non-local host (SYN and FIN packets).

For a complete description of the capture filters please see the following link: www.ethereal.com/tcpdump.8.html.

Display filters allow you to filter for such things as a specific value in a certain protocol, or to check and see if a certain field or protocol was captured. One simple example would be to use **udp** to display any udp traffic. Fields, using English abbreviations or C programming symbols, can be compared against values. Some examples would be equal (eq or ==), not equal (ne or !=), and greater than (gt or >). You can also use logical expressions to combine filters. Some logical expressions are logical AND (and or &&), logical NOT (not or !), and logical OR (or or | |). It would take some time to explain every nuance, so we'll look at some examples. For an in-depth look at display filters please see www.ethereal.com/ethereal.1.html.

ip.addr == 192.168.0.0/16- Would show all packets in the 192.168.0.0 B class network.

tcp.port == 5190 && tcp.port == 21- Show all packets using AOL Instant Messenger and the File Transfer Protocol.

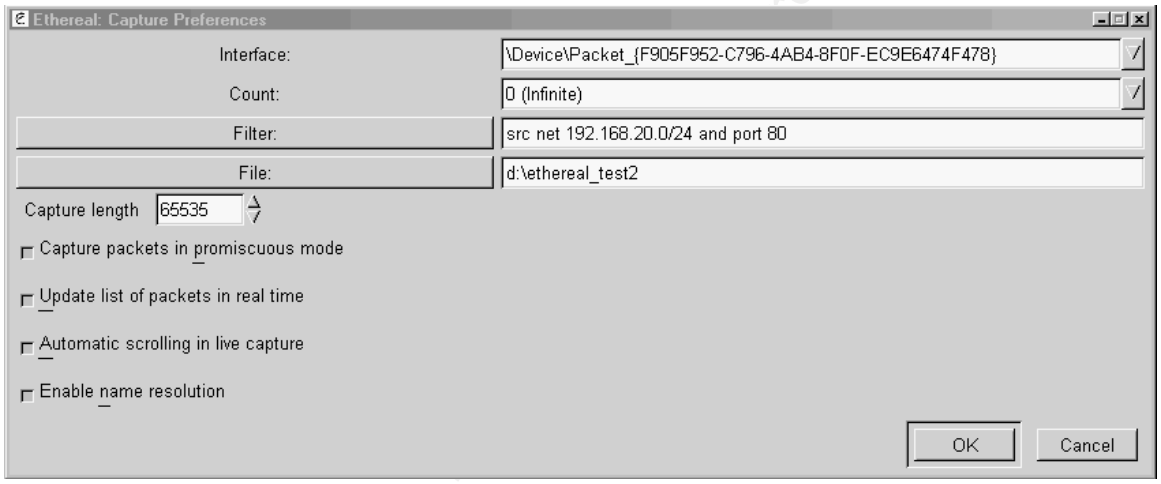
! (ip.addr 10.1.1.138 or ipx)- Show all packets except those from host 10.1.1.138 and leave out any ipx traffic as well.

Now that we have a basic understanding of filters, let's take a look at some real examples of what Ethereal can do for you.

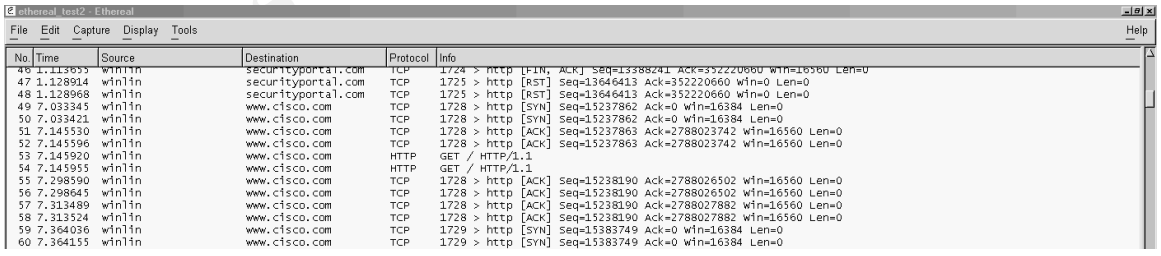
```
69 2.998392 192.168.20.55       192.168.20.101      FTP     Response: 220 ultra Microsoft FTP Service (Version 4.0).
70 2.998474 192.168.20.101      192.168.20.55       FTP     Request: USER bsmith
71 2.998503 192.168.20.101      192.168.20.55       FTP     Request: USER bsmith
72 2.998839 192.168.20.55       192.168.20.101      FTP     Response: 331 Password required for bsmith.
73 2.998920 192.168.20.101      192.168.20.55       FTP     Request: PASS bob
74 2.998947 192.168.20.101      192.168.20.55       FTP     Request: PASS bob
75 2.999751 192.168.20.55       192.168.20.101      FTP     Response: 230-Welcome to the TEST FTP Site
76 3.158249 192.168.20.101      192.168.20.55       TCP     1718 > 21 [ACK] Seq=4117896007 Ack=13385089 Win=17112 Len=0
77 3.158328 192.168.20.101      192.168.20.55       TCP     1718 > 21 [ACK] Seq=4117896007 Ack=13385089 Win=17112 Len=0
78 3.158409 192.168.20.101      192.168.20.55       TCP     1719 > 21 [ACK] Seq=4117982725 Ack=13384978 Win=17403 Len=0
79 3.158428 192.168.20.101      192.168.20.55       TCP     1719 > 21 [ACK] Seq=4117982725 Ack=13384978 Win=17403 Len=0
80 3.158680 192.168.20.55       192.168.20.101      FTP     Response: 230 User bsmith logged in.
81 3.158890 192.168.20.101      192.168.20.55       FTP     Request: opts utf8 on
82 3.158919 192.168.20.101      192.168.20.55       FTP     Request: opts utf8 on
83 3.159295 192.168.20.55       192.168.20.101      FTP     Response: 500 'OPTS utf8 on': command not understood
84 3.159370 192.168.20.101      192.168.20.55       FTP     Request: syst
```

Above we have output from Ethereal showing an FTP session. Notice anything interesting? YES, you can see the username and password in plain text. Quite handy if you wanted to monitor unauthorized attempts to your FTP server.

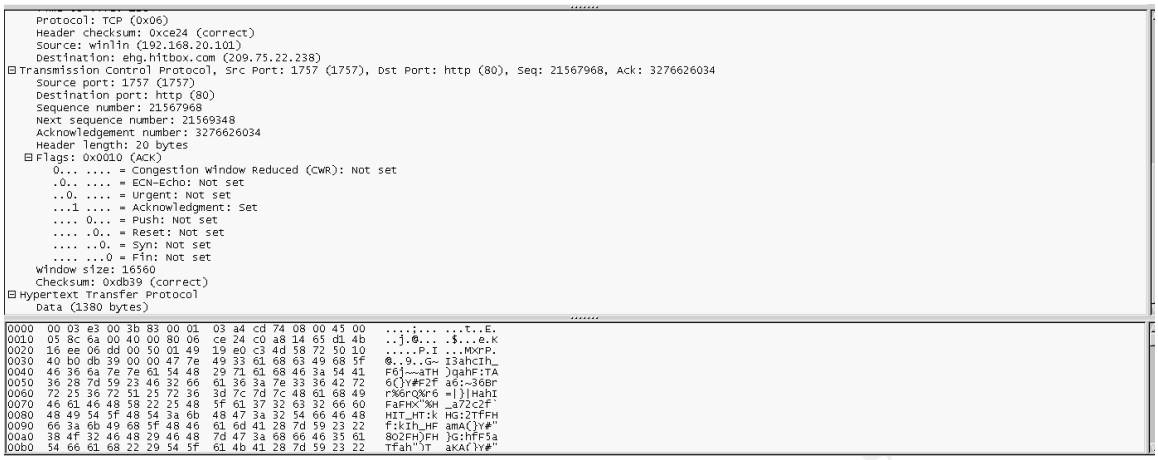
Now suppose you would like to monitor Internet traffic from a specific network because of some complaints. Here's what the capture filter would look like:



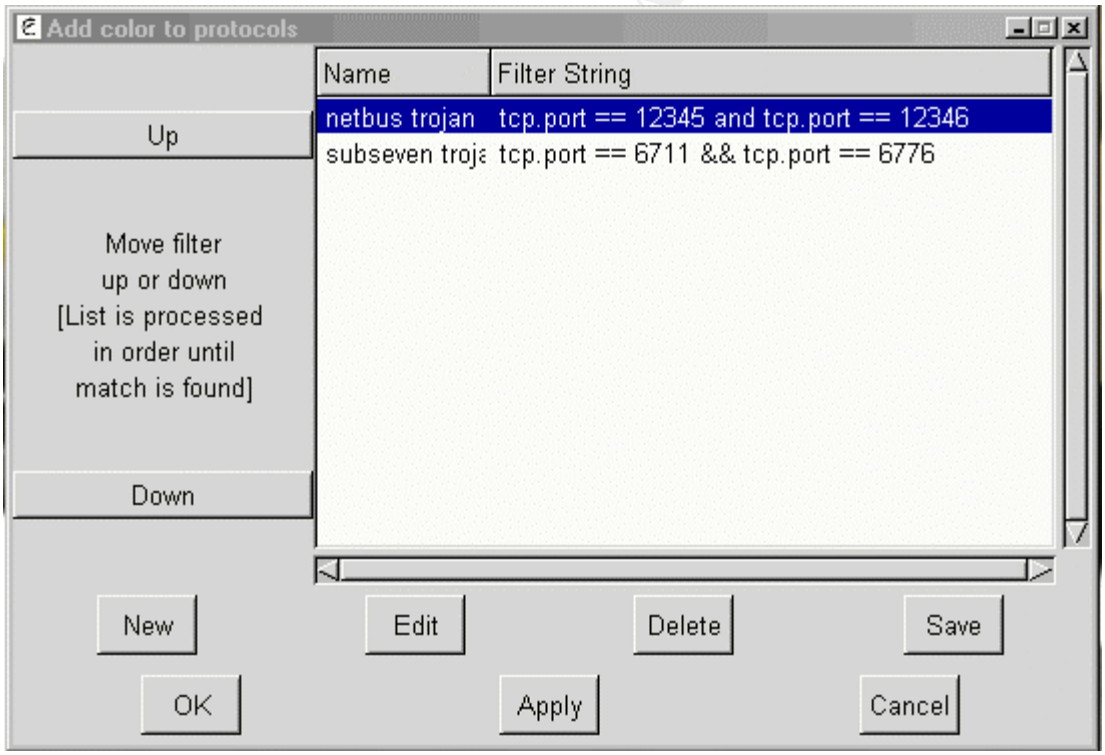
Our source network is the 192.168.20.0 net, and we want only port 80 traffic. Also checked is the enable name resolution, to make it easier to view actual site names. You could be more specific in the filter if you wished by adding the destination of a forbidden site. And here's part of the output:



Ethereal has two other panes that can give you extensive information about the actual packets, as well as a hex dump as shown below:



One way to filter traffic that you are looking for is to highlight a field in one of the windows. You can then go up to the menu and click on Display, and then choose Match Selected. Ethereal will show you packets that contain your selection. Another convenient feature allows you to colorize your filters. Say for example that you were looking for NetBus and SubSeven traffic on your network. You could create a color filter for the ports that these trojan viruses use.



Yet another way to use Ethereal to as an added layer of defense-in-depth would be to get yourself a file that contained packet information for say a Denial of Service attack. You could use Ethereal's powerful capture filtering capabilities. Let's say that you wanted to watch your old Windows NT IIS server for Snork attacks. You could write a capture filter that looked like this: dst host 192.168.1.10 && dst port 135 && (src port 7 or 9 or 135). This particular filter says to capture traffic going to IP address 192.168.1.10 (IIS server) and that has a destination port of 135 (tcp or udp) and a source port of 7, 9, or 135.

The best way to put Ethereal to the test, however, is to experiment with display and capture filters. You'll eventually learn what you need to filter, but make sure you've established a baseline of your network traffic first. It will make writing filters easier, when you know what you don't need to look at.

SUMMARY

Ethereal is an excellent packet-capturing tool that will allow you to see just what's traveling over your network. With the proper filters and, Ethereal could act in conjunction with your IDS system to give you more coverage of your network. As always, please have written and signed permission before you start sniffing traffic.

CONCLUSION

I thank you for your time, and hope that you are ready to add Namp and Ethereal to your security toolkit. Both tools will greatly aid in the battle against those who mean to do your network and its information harm. A small investment in the time that it takes to learn to use both tools will reap satisfying results in the immediate future.

There are many other free security tools available on the Internet. Your favorite search engine will help get the job done.

---

[1] McClure, Stuart and Scambray, Joel. "Free Windows-based scanners are plentiful, but only Asmodeus shows promise." InfoWorld 6 July 1998.
URL: http://www.infoworld.com/cgi-bin/displayNew.pl?/security/980706sw.htm

[2] NmapFE is available at the following link as a RedHat Program Manager (RPM) install.
URL: http://www.insecure.org/nmap/dist/nmap-frontend-0.2.53-1.i386.rpm

[3] Nmap is bundled with NmapFE as a regular gzip file.
URL: http://www.insecure.org/nmap/dist/nmap-2.53.tgz

[4] An article on OS fingerprinting.
URL: http://www.insecure.org/nmap/nmap-fingerprinting-article.html

[5] Graham, Robert. "Sniffing (network wiretap, sniffer) FAQ".
URL: http://www.robertgraham.com/pubs/sniffing-faq.html

Appendix A

** This was derived using the netstat -t •u command**

Active Internet connections (servers and established)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	29	pool-63.50.x.x:sunrpc	64.9.x.x:16527	LAST_ACK
tcp	0	0	pool-63.50.x.x:1032	64.9.x.x:1352	CLOSE
tcp	0	0	*:20337	*:*	LISTEN
tcp	0	0	think:1024	*:*	LISTEN
tcp	0	0	pool-63.50.x.x.kgp:X	64.9.x.x:16508	SYN_RECV
tcp	0	0	pool-63.50.x.x.kgp:X	64.9.x.x:16509	SYN_RECV
tcp	0	0	pool-63.50.x.x.kgp:X	64.9.x.x:16511	SYN_RECV
tcp	0	0	*:X	*:*	LISTEN
tcp	0	0	pool-63.50.x.x.k:www	64.9.x.x:16509	SYN_RECV
tcp	0	0	pool-63.50.x.x.k:www	64.9.x.x:16511	SYN_RECV
tcp	0	0	pool-63.50.x.x.k:www	64.9.x.x:16508	SYN_RECV
tcp	0	0	*:www	*:*	LISTEN
tcp	0	0	pool-63.50.x.x:https	64.9.x.x:16508	SYN_RECV
tcp	0	0	pool-63.50.x.x:https	64.9.x.x:16509	SYN_RECV
tcp	0	0	pool-63.50.x.x:https	64.9.x.x:16511	SYN_RECV
tcp	0	0	*:https	*:*	LISTEN
tcp	0	0	pool-63.50.x.x:linuxconf	64.9.x.x:16508	SYN_RECV
tcp	0	0	pool-63.50.x.x:linuxconf	64.9.x.x:16509	SYN_RECV
tcp	0	0	pool-63.50.x.x:linuxconf	64.9.x.x:16511	SYN_RECV
tcp	0	0	*:linuxconf	*:*	LISTEN
tcp	0	0	pool-63.50.x.x:login	64.9.x.x:16508	SYN_RECV
tcp	0	0	pool-63.50.x.x:login	64.9.x.x:16509	SYN_RECV
tcp	0	0	pool-63.50.x.x:login	64.9.x.x:16512	SYN_RECV
tcp	0	0	pool-63.50.x.x:login	64.9.x.x:16513	SYN_RECV
tcp	0	0	pool-63.50.x.x:login	64.9.x.x:16515	SYN_RECV
tcp	0	0	*:login	*:*	LISTEN
tcp	0	0	pool-63.50.x.x:shell	64.9.x.x:16512	SYN_RECV
tcp	0	0	pool-63.50.x.x:shell	64.9.x.x:16513	SYN_RECV
tcp	0	0	pool-63.50.x.x:shell	64.9.x.x:16515	SYN_RECV
tcp	0	0	*:shell	*:*	LISTEN
tcp	0	0	pool-63.50.x.x:telnet	64.9.x.x:16508	SYN_RECV
tcp	0	0	pool-63.50.x.x:telnet	64.9.x.x:16509	SYN_RECV

```
tcp        0      0 pool-63.50.x.x:telnet 64.9.x.x:16511       SYN_RECV
tcp        0      0 *:telnet                *:*                    LISTEN
tcp        0      0 pool-63.50.x.x.k:ftp 64.9.x.x:16508       SYN_RECV
tcp        0      0 pool-63.50.x.x.k:ftp 64.9.x.x:16509       SYN_RECV
tcp        0      0 pool-63.50.x.x.k:ftp 64.9.x.x:16511       SYN_RECV
tcp        0      0 *:ftp                   *:*                    LISTEN
tcp        0      0 pool-63.50.x.x.k:ssh 64.9.x.x:16508       SYN_RECV
tcp        0      0 pool-63.50.x.x.k:ssh 64.9.x.x:16509       SYN_RECV
tcp        0      0 pool-63.50.x.x.k:ssh 64.9.x.x:16511       SYN_RECV
tcp        0      0 *:ssh                   *:*                    LISTEN
tcp        0      0 pool-63.50.x.x.:auth 64.9.x.x:16508       SYN_RECV
tcp        0      0 pool-63.50.x.x.:auth 64.9.x.x:16509       SYN_RECV
tcp        0      0 pool-63.50.x.x.:auth 64.9.x.x:16511       SYN_RECV
tcp        0      0 *:auth                  *:*                    LISTEN
tcp        0      0 pool-63.50.x.x:sunrpc 64.9.x.x:16509      SYN_RECV
tcp        0      0 pool-63.50.x.x:sunrpc 64.9.x.x:16511      SYN_RECV
tcp        0      0 *:sunrpc                *:*                    LISTEN
udp        0      0 think:1059              think:1059          ESTABLISHED
udp        0      0 *:tftp                  *:*
udp        0      0 *:sunrpc                *:*
```

some interesting output from /var/log/messages

```
May 11 12:53:53 thinkunix sshd[2456]: Did not receive ident string from 64.9.x.x.
May 11 12:53:56 thinkunix telnetd[2457]: ttloop: peer died: EOF
May 11 12:54:07 thinkunix xinetd[412]: warning: can't get client address: Invalid argument
May 11 12:54:20 thinkunix ftpd[2455]: FTP session closed
May 11 12:55:08 thinkunix rshd[2461]: Connection from 64.9.x.x on illegal port
```

Last Updated: September 20th, 2017

Upcoming Training

Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS New York SEC401*	New York, NY	Sep 25, 2017 - Sep 30, 2017	Community SANS
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Community SANS Sacramento SEC401	Sacramento, CA	Oct 02, 2017 - Oct 07, 2017	Community SANS
Mentor Session - SEC401	Minneapolis, MN	Oct 03, 2017 - Nov 14, 2017	Mentor
Mentor Session - SEC401	Arlington, VA	Oct 04, 2017 - Nov 15, 2017	Mentor
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZ	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VA	Oct 14, 2017 - Oct 21, 2017	Live Event
CCB Private SEC401 Oct 17	Brussels, Belgium	Oct 16, 2017 - Oct 21, 2017	
SANS Tokyo Autumn 2017	Tokyo, Japan	Oct 16, 2017 - Oct 28, 2017	Live Event
Community SANS Omaha SEC401	Omaha, NE	Oct 23, 2017 - Oct 28, 2017	Community SANS
SANS vLive - SEC401: Security Essentials Bootcamp Style	SEC401 - 201710,	Oct 23, 2017 - Nov 29, 2017	vLive
San Diego Fall 2017 - SEC401: Security Essentials Bootcamp Style	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, United Arab Emirates	Nov 04, 2017 - Nov 16, 2017	Live Event
Community SANS Colorado Springs SEC401**	Colorado Springs, CO	Nov 06, 2017 - Nov 11, 2017	Community SANS
SANS Miami 2017	Miami, FL	Nov 06, 2017 - Nov 11, 2017	Live Event
Community SANS Vancouver SEC401*	Vancouver, BC	Nov 06, 2017 - Nov 11, 2017	Community SANS
SANS Sydney 2017	Sydney, Australia	Nov 13, 2017 - Nov 25, 2017	Live Event
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS St. Louis SEC401	St Louis, MO	Nov 27, 2017 - Dec 02, 2017	Community SANS
Community SANS Portland SEC401	Portland, OR	Nov 27, 2017 - Dec 02, 2017	Community SANS
SANS San Francisco Winter 2017	San Francisco, CA	Nov 27, 2017 - Dec 02, 2017	Live Event
SANS London November 2017	London, United Kingdom	Nov 27, 2017 - Dec 02, 2017	Live Event
SANS Khobar 2017	Khobar, Saudi Arabia	Dec 02, 2017 - Dec 07, 2017	Live Event
SANS Austin Winter 2017	Austin, TX	Dec 04, 2017 - Dec 09, 2017	Live Event