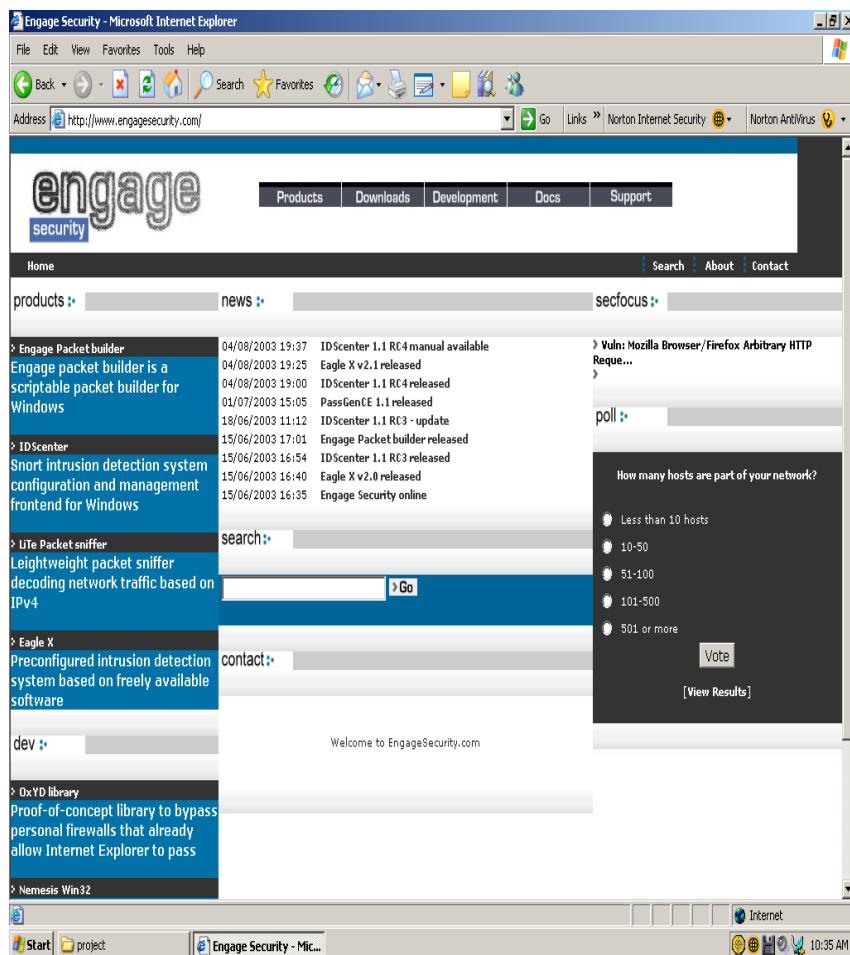


# Packet Testing Tools

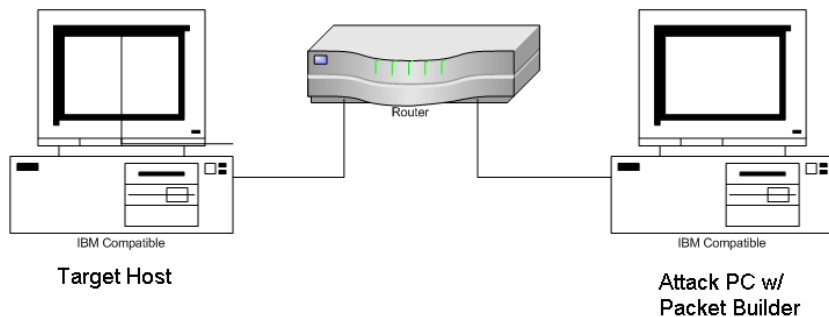
By Dan Dirks

# Engage Packet Builder



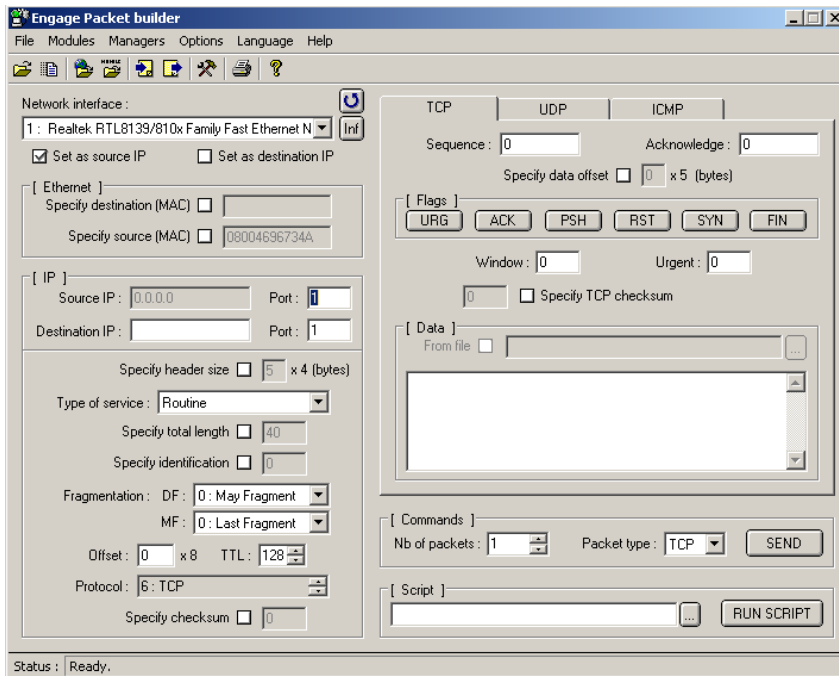
- Engage packet builder is a scriptable packet builder for Windows
- Freeware
- Found at [www.engagesecurity.com](http://www.engagesecurity.com)
- Uses WinPcap 2.3( included on the download package)
- Very versatile and its use is mostly intuitive
- Example scripts are included
  
- **CAUTION:**
- WinPcap is a popular Open Source Windows packet library used in many network tools including Ethereal. The current version of WinPcap is 3.1. Unfortunately Packet Builder is not compatible with version 3.1. WinPcap 3.1 is currently installed on the PC it must be uninstalled before installing Packet Builder with WinPcap 2.3
- Packet builder comes with very little documentation; a handful of sample scripts and a forum on the webpage.

# Test set up



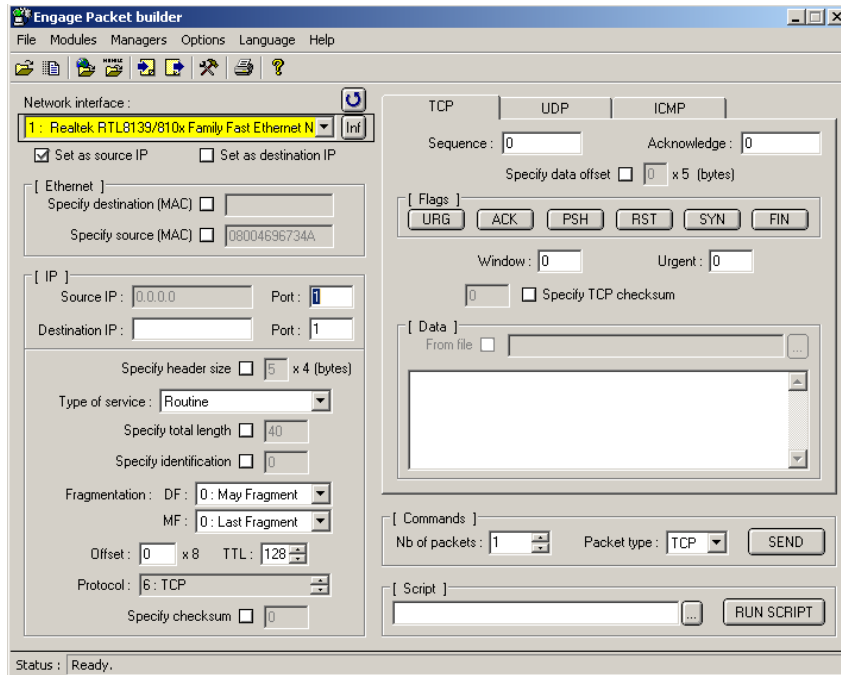
- Using packet Builder to test a router or firewall IDS or ACL
- Requires two PCs.
- The setup may be modified without the router to run PC to PC.

# Opening Screen



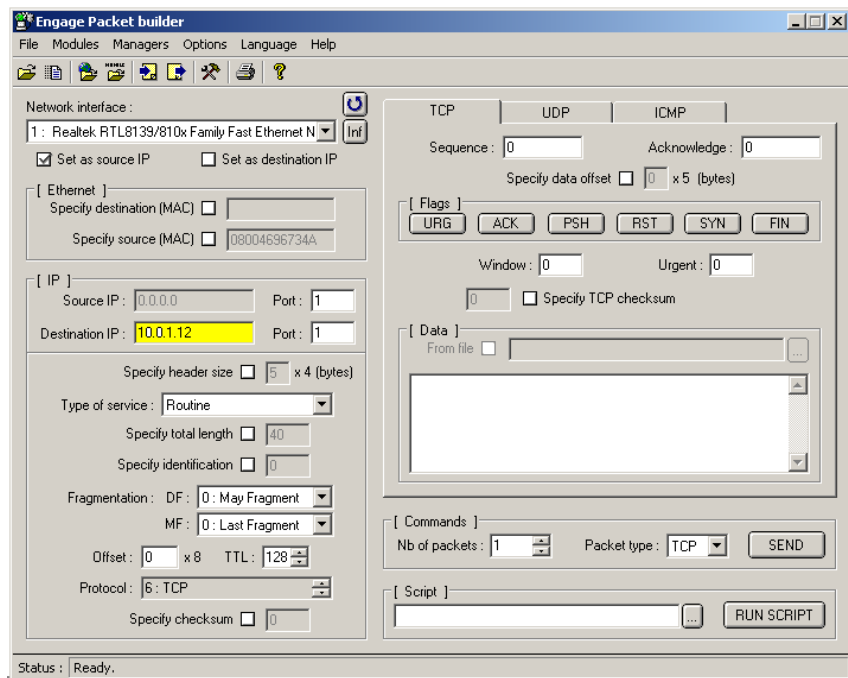
- The left side contains the network interface, Ethernet and IP options.
- The right side contains the packet type tabs TCP, UDP and ICMP.
- The bottom right contains the script options.

# Choose the interface



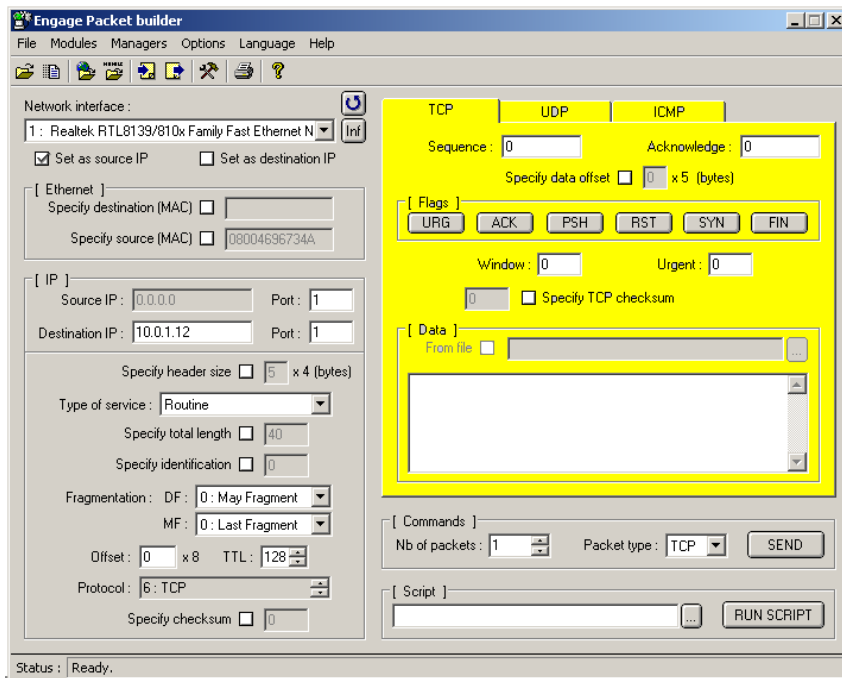
- If the NIC is up the program will automatically sense the hardware and the MAC.
- But you still have to choose the NIC card from the pulldown menu

# Input the target IP address



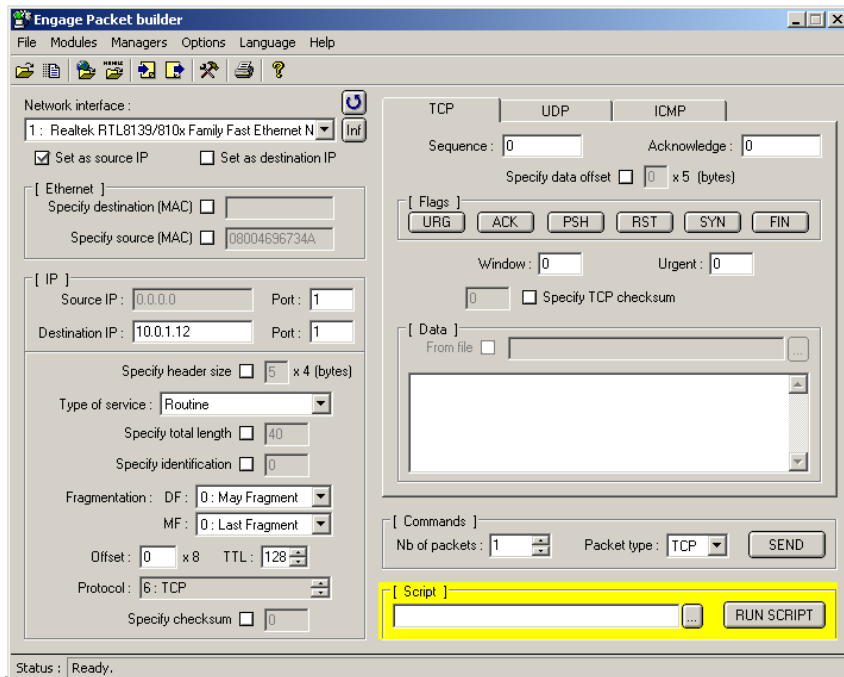
- A target IP address is required, the port number may need to be specified depending upon the packet type you want
- All other IP options may remain as default

# Build the packet



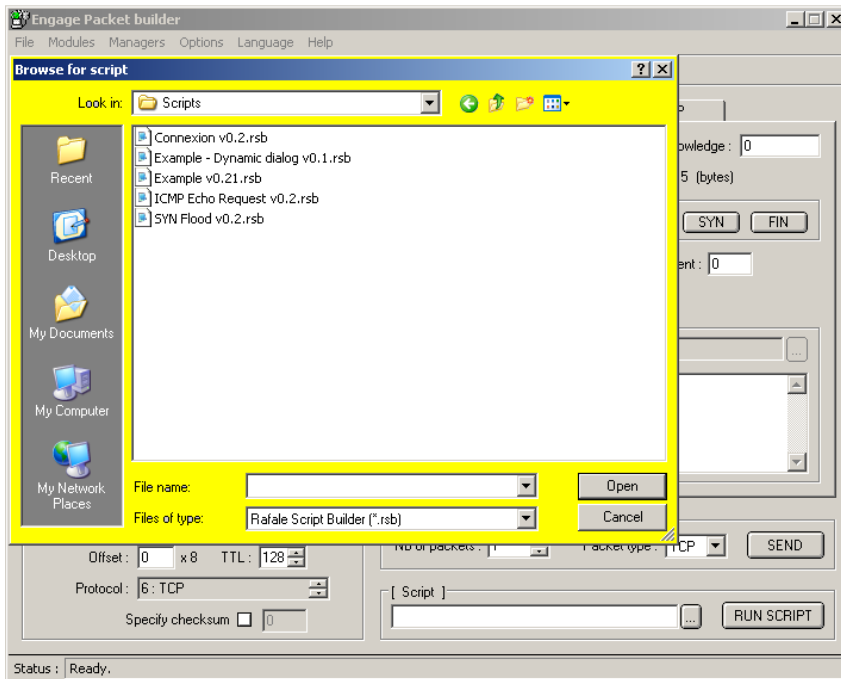
- Choose the type by clicking the one of the tabs
- Build the packet by choosing the options
- Once the packet is built, click send. Notice that you can choose the number of packets to send.

# OR you can run a script



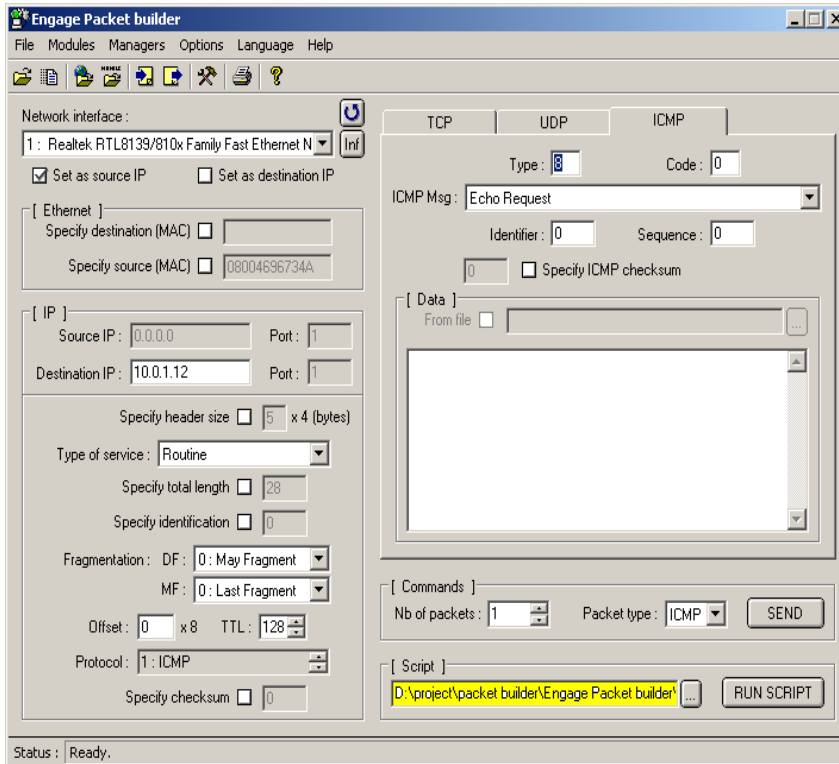
- Hit the ellipses button to bring up the script directory

# Choose the script you want



- Let's choose the ICMP echo request
- A simple ping test

# Ready to send the script



- We are ready to run the script; however, Packet Builder only transmits.
- We need a way to see if the target host receives the packets.
- So before sending, set up the target host to show if the packets are received.

# On the Target host

```
C:\WINNT\system32\cmd.exe
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\Documents and Settings\Administrator>netstat ?

Displays protocol statistics and current TCP/IP network connections.

NETSTAT [-a] [-e] [-n] [-s] [-p proto] [-r] [interval]

-a          Displays all connections and listening ports.
-e          Displays Ethernet statistics. This may be combined with the -s
           option.
-n          Displays addresses and port numbers in numerical form.
-p proto    Shows connections for the protocol specified by proto; proto
           may be TCP or UDP. If used with the -s option to display
           per-protocol statistics, proto may be TCP, UDP, or IP.
-r          Displays the routing table.
-s          Displays per-protocol statistics. By default, statistics are
           shown for TCP, UDP and IP; the -p option may be used to specify
           a subset of the default.
interval   Redisplays selected statistics, pausing interval seconds
           between each display. Press CTRL+C to stop redisplaying
           statistics. If omitted, netstat will print the current
           configuration information once.

C:\Documents and Settings\Administrator>
```

- Open a command prompt
- Use the 'netstat' command with the -s and -p ICMP parameters

# netstat

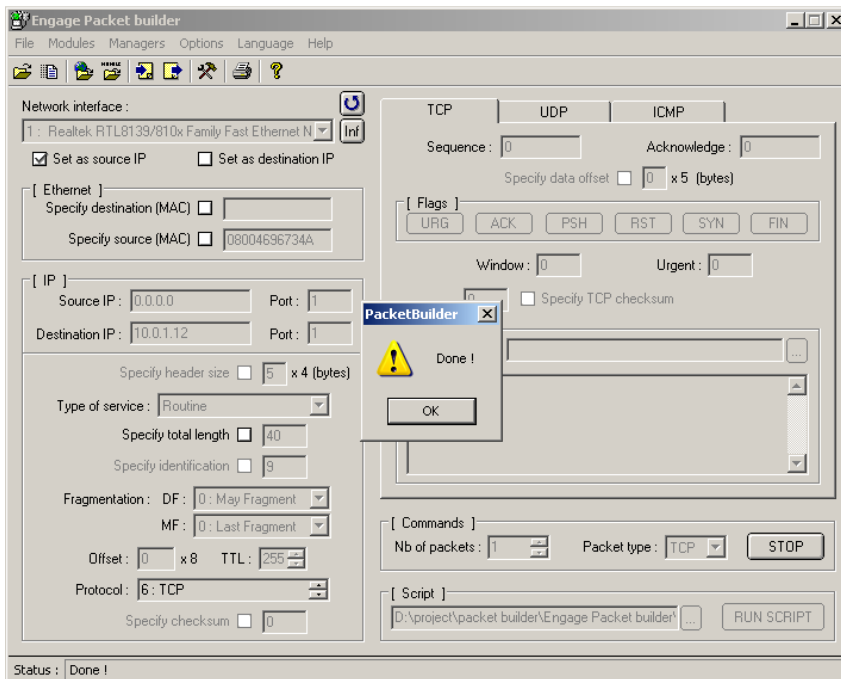
```
C:\Documents and Settings\Administrator>netstat -s -p icmp
ICMP Statistics

      Received      Sent
Messages          0          0
Errors             0          0
Destination Unreachable 0          0
Time Exceeded     0          0
Parameter Problems 0          0
Source Quenches   0          0
Redirects         0          0
Echos             0          0
Echo Replies      0          0
Timestamps       0          0
Timestamp Replies 0          0
Address Masks     0          0
Address Mask Replies 0          0

C:\Documents and Settings\Administrator>
```

- The `-s` parameter show the statistics
- The `-p ICMP` parameter filters out all but ICMP stats
- Currently shows 0 packets sent, 0 packets received

# Run the script from the attack PC



- Click the Run Script button
- The form grays out and a done text box appears when the packets are sent
- Click OK

# Back to the target host

```
C:\Documents and Settings\Administrator>netstat -s -p icmp
```

## ICMP Statistics

	Received	Sent
Messages	0	0
Errors	0	0
Destination Unreachable	0	0
Time Exceeded	0	0
Parameter Problems	0	0
Source Quenches	0	0
Redirects	0	0
Echos	0	0
Echo Replies	0	0
Timestamps	0	0
Timestamp Replies	0	0
Address Masks	0	0
Address Mask Replies	0	0

```
C:\Documents and Settings\Administrator>netstat -s -p icmp
```

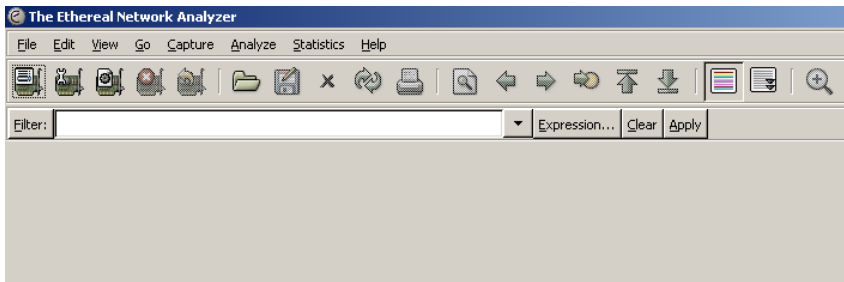
## ICMP Statistics

	Received	Sent
Messages	3	0
Errors	3	0
Destination Unreachable	0	0
Time Exceeded	0	0
Parameter Problems	0	0
Source Quenches	0	0
Redirects	0	0
Echos	0	0
Echo Replies	0	0
Timestamps	0	0
Timestamp Replies	0	0
Address Masks	0	0
Address Mask Replies	0	0

```
C:\Documents and Settings\Administrator>_
```

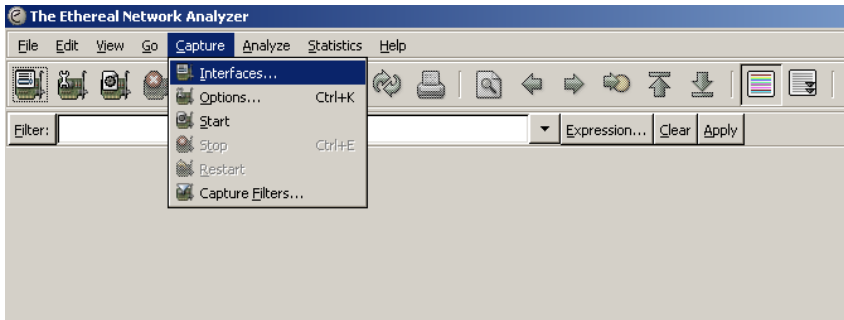
- Run the netstat -s -p ICMP command again
- Compare to the previous stats:
- 3 messages received
- 3 errors recorded
- The 'ping' failed, but why?

# Launch Ethereal



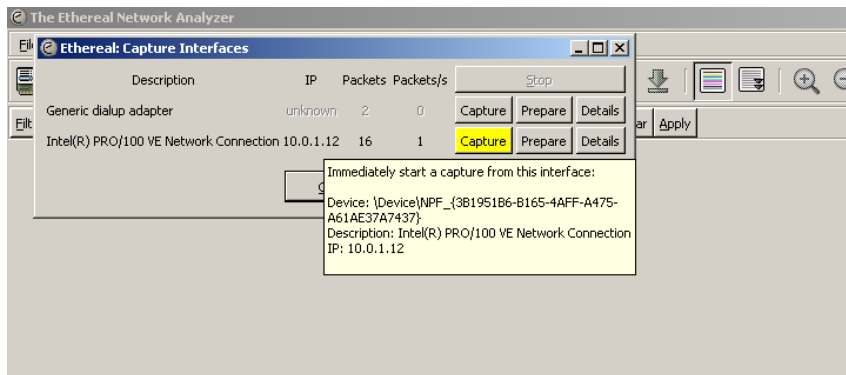
- To examine the packets being received, use Ethereal on the Target PC, another freeware program.
- [www.ethereal.com](http://www.ethereal.com)
- We need to capture packets received by the target when we run the script again

# Select the interface



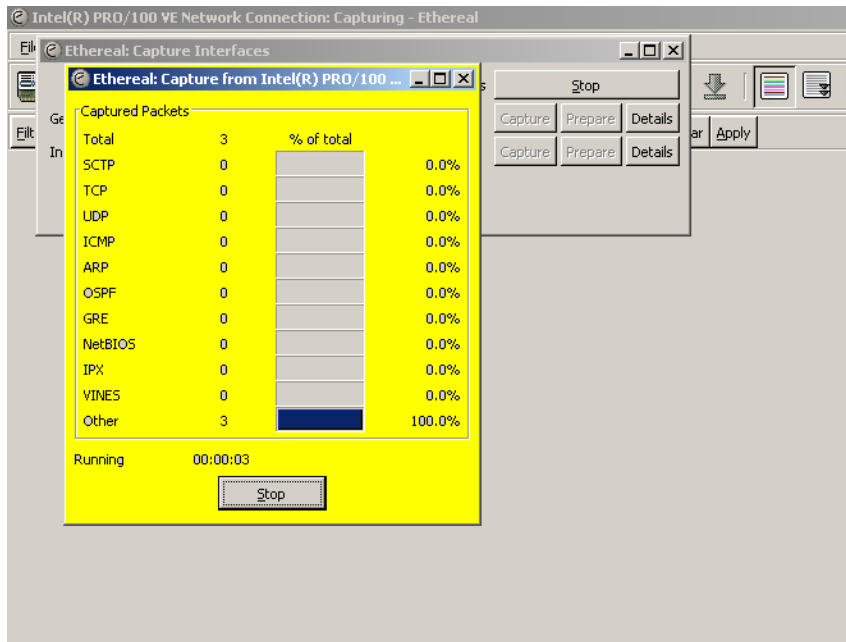
- To capture the packets; select 'Capture', then select 'Interfaces'

# Start the capture



- Click the capture button along side the ethernet interface

# Capture running



- With the capture now running, return to the attack PC and run the script again.
- When the script run completes, return to Ethereal on the target PC and click stop

# Ethereal capture screen

The screenshot shows the Wireshark interface with a packet capture. The main pane displays a list of packets with columns for No., Time, Source, Destination, Protocol, and Info. Packet 53 is selected and highlighted in dark blue. Below the list, the packet details pane shows the structure of the selected packet: Frame 53 (60 bytes on wire, 60 bytes captured), Ethernet II, Internet Protocol, and Internet Control Message Protocol. The packet bytes pane shows the raw hex and ASCII data for the selected packet.

No.	Time	Source	Destination	Protocol	Info
47	62.344619	10.0.1.2	224.0.0.10	EIGRP	Hello
50	66.872585	10.0.1.2	224.0.0.10	EIGRP	Hello
58	71.680536	10.0.1.2	224.0.0.10	EIGRP	Hello
53	70.298762	10.0.2.10	10.0.1.12	ICMP	Echo (ping) request
54	70.299317	10.0.2.10	10.0.1.12	ICMP	Echo (ping) request
55	70.299901	10.0.2.10	10.0.1.12	ICMP	Echo (ping) request
56	70.300502	10.0.2.10	10.0.1.12	ICMP	Echo (ping) request
57	70.301076	10.0.2.10	10.0.1.12	ICMP	Echo (ping) request
1	0.000000	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
3	2.003741	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
4	4.003423	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
5	6.006214	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
7	8.009026	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
8	10.011828	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
10	12.014656	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
11	14.017493	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
12	16.020241	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d

Frame 53 (60 bytes on wire, 60 bytes captured)  
Ethernet II, Src: Cisco\_8f:d0:a8 (00:14:69:8f:d0:a8), Dst: Intel\_b5:d6:29 (00:07:e9:b5:d6:29)  
Internet Protocol, Src: 10.0.2.10 (10.0.2.10), Dst: 10.0.1.12 (10.0.1.12)  
Internet Control Message Protocol

```
0000 00 07 e9 b5 d6 29 00 14 69 8f d0 a8 08 00 45 00  ....).f.....E.  
0010 00 1c 00 09 00 00 fc 01 a7 c2 0a 00 02 0a 0a 00  .....>.....  
0020 01 0c 08 00 19 3e 03 00 01 00 00 00 00 00 00 00  .....>.....  
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
```

- When you stop the capture, the packets captured are listed
- Click the 'Protocol' column to sort the list
- Select the first ICMP packet listed (dark blue line)
- Notice the middle window below the list
- Notice the 4 items listed in the middle window (Frame, Ethernet II, Internet Protocol, and Internet Control Message Protocol) and the plus icon to the left of each item

# Examine the first ICMP packet

The screenshot shows the Wireshark interface with a list of captured packets. The first ICMP packet (No. 53) is highlighted in yellow. The details pane for this packet is expanded, showing the following information:

- Arrival Time: Oct 4, 2005 16:18:36.117654000
- [Time delta from previous packet: 0.200856000 seconds]
- [Time since reference or first frame: 70.298762000 seconds]
- Frame Number: 53
- Packet Length: 60 bytes
- Capture Length: 60 bytes
- [Protocols in frame: eth:ip:icmp:data]
- Ethernet II, Src: Cisco\_8f:d0:a8 (00:14:69:8f:d0:a8), Dst: Intel\_b5:d6:29 (00:07:e9:b5:d6:29)
- Internet Protocol, Src: 10.0.2.10 (10.0.2.10), Dst: 10.0.1.12 (10.0.1.12)
- Internet Control Message Protocol

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```
0000  00 07 e9 b5 d6 29 00 14 69 8f d0 a8 08 00 45 00  ....). 1....E.
0010  00 1c 00 09 00 00 fc 01 a7 c2 0a 00 02 0a 0a 00
0020  01 0c 08 00 19 3e 03 00 01 00 00 00 00 00 00 00
0030  00 00 00 00 00 00 00 00 00 00 00 00
```

- You can examine the packet details in the middle (highlighted yellow) window by clicking the plus icons on the left
- This shows the details of the packet frame
- The errors seen by Netstat don't appear to be in the Frame.

# Ethernet details

The screenshot displays the Wireshark interface with the following details:

- Filter:** Expression... Clear Apply
- Packet List:**

No.	Time	Source	Destination	Protocol	Info
47	62.344619	10.0.1.2	224.0.0.10	EIGRP	Hello
50	66.872585	10.0.1.2	224.0.0.10	EIGRP	Hello
58	71.680536	10.0.1.2	224.0.0.10	EIGRP	Hello
53	70.298762	10.0.2.10	10.0.1.12	ICMP	Echo (ping) request
54	70.299317	10.0.2.10	10.0.1.12	ICMP	Echo (ping) request
55	70.299901	10.0.2.10	10.0.1.12	ICMP	Echo (ping) request
56	70.300502	10.0.2.10	10.0.1.12	ICMP	Echo (ping) request
57	70.301076	10.0.2.10	10.0.1.12	ICMP	Echo (ping) request
1	0.000000	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
3	2.003741	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
4	4.003423	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
5	6.006214	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
7	8.009026	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
8	10.011828	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
10	12.014656	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
11	14.017493	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
12	16.020241	Cisco_c7:ae:81	Spanning-tree-(for STP	Conf.	Root = 32768/00:d0:ba:c7:ae:85 Cost = 0 Port = 0x800d
- Packet 53 Details:**
  - Frame 53 (60 bytes on wire, 60 bytes captured)
  - Ethernet II, Src: Cisco\_8f:d0:a8 (00:14:69:8f:d0:a8), Dst: Intel\_b5:d6:29 (00:07:e9:b5:d6:29)
    - Destination: Intel\_b5:d6:29 (00:07:e9:b5:d6:29)
    - Source: Cisco\_8f:d0:a8 (00:14:69:8f:d0:a8)
    - Type: IP (0x0800)
    - Trailer: 00000000000000000000000000000000
  - Internet Protocol, Src: 10.0.2.10 (10.0.2.10), Dst: 10.0.1.12 (10.0.1.12)
  - Internet Control Message Protocol
- Packet Bytes:**

```
0000 00 07 e9 b5 d6 29 00 14 69 8f d0 a8 08 00 45 00 .....).I....E.
0010 00 1c 00 09 00 00 fc 01 a7 c2 0a 00 02 0a 0a 00 .....
0020 01 0c 08 00 19 3e 03 00 01 00 00 00 00 00 00 00 .....>.....
0030 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

- This shows the Ethernet details (highlighted yellow)
- The errors don't appear here either

# IP Details

The screenshot shows the Wireshark interface with the following details:

No.	Time	Source	Destination	Protocol	Info
47	62.344619	10.0.1.2	224.0.0.10	EIGRP	Hello
50	66.872585	10.0.1.2	224.0.0.10	EIGRP	Hello
58	71.680536	10.0.1.2	224.0.0.10	EIGRP	Hello
53	70.298762	10.0.2.10	10.0.1.12	ICMP	Echo (ping) request
54	70.299317	10.0.2.10	10.0.1.12	ICMP	Echo (ping) request

Frame 53 (60 bytes on wire, 60 bytes captured)  
Ethernet II, Src: Cisco\_8f:d0:a8 (00:14:69:8f:d0:a8), Dst: Intel\_b5:d6:29 (00:07:e9:b5:d6:29)  
Internet Protocol, Src: 10.0.2.10 (10.0.2.10), Dst: 10.0.1.12 (10.0.1.12)

Version: 4  
Header length: 20 bytes  
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)  
0000 00.. = Differentiated Services Codepoint: Default (0x00)  
.... ..0. = ECN-Capable Transport (ECT): 0  
.... ..0 = ECN-CE: 0  
Total Length: 28  
Identification: 0x0009 (9)  
Flags: 0x00  
0... = Reserved bit: Not set  
.0.. = Don't fragment: Not set  
..0. = More fragments: Not set  
Fragment offset: 0  
Time to live: 252  
Protocol: ICMP (0x01)  
Header checksum: 0xa7c2 [correct]  
source: 10.0.2.10 (10.0.2.10)  
destination: 10.0.1.12 (10.0.1.12)  
Internet Control Message Protocol

0000 00 07 e9 b5 d6 29 00 14 69 8f d0 a8 08 00 45 00 ..... 1.....E.  
0010 00 1c 00 09 00 00 fc 01 a7 c2 0a 00 02 0a 0a 00 .....>.....  
0020 01 0c 08 00 19 3e 03 00 01 00 00 00 00 00 00 00 .....  
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

- This shows the IP details (highlighted yellow)
- And the IP seems to be error free

# ICMP Detail

(Untitled) - Ethereal

File Edit View Go Capture Analyze Statistics Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
47	62.344619	10.0.1.2	224.0.0.10	EIGRP	Hello
50	66.872585	10.0.1.2	224.0.0.10	EIGRP	Hello
58	71.680536	10.0.1.2	224.0.0.10	EIGRP	Hello
53	70.298762	10.0.2.10	10.0.1.12	ICMP	Echo (ping) request
54	70.299317	10.0.2.10	10.0.1.12	ICMP	Echo (ping) request

Frame 53 (60 bytes on wire, 60 bytes captured)

- Ethernet II, Src: Cisco\_8f:d0:a8 (00:14:69:8f:d0:a8), Dst: Intel\_b5:d6:29 (00:07:e9:b5:d6:29)
- Internet Protocol, Src: 10.0.2.10 (10.0.2.10), Dst: 10.0.1.12 (10.0.1.12)
- Internet Control Message Protocol
  - Type: 8 (Echo (ping) request)
  - Code: 0
  - Checksum: 0x193e [incorrect, should be 0xf3ff]
  - Identifier: 0x0300
  - Sequence number: 0x0100

```
0000  00 07 e9 b5 d6 29 00 14 69 8f d0 a8 08 00 45 00  .....). 1.....E.
0010  00 1c 00 09 00 00 fc 01 a7 c2 0a 00 02 0a 0a 00  .....>.....
0020  01 0c 08 00 19 3e 05 00 01 00 00 00 00 00 00 00  .....>.....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
```

Internet Control Message Protocol (icmp) | P: 59 D: 59 M: 0 Drops: 0

Start | router 1 - Hyp... | F:\screens | C:\WINNT\sys... | untitled - Paint | (Untitled) - E... | 4:27 PM

- Ah Ha!
- Packet Builder is sending the wrong ICMP checksum (highlighted yellow)
- Well it's freeware and you get what you pay for.
- But is it the program or the script?

# Back to the Attack PC

```
• // Rafale X script
• // -----
• // Action : Turn off vulnerable modems
• //
• %name=ICMP Echo Request v0.2
• %category=Test
• %date=02/06/2002
• %packetbuildermin=0.2
• %description=Turn off vulnerable modems.
• %description=(ex: Olitec SpeedCom)

• // Variables
• $done=Done !
• $packets=5

• // Do the stuff...
• IPOFFSET=0
• IPIDENT=9
• IPTTL=255

• ICMPTYPE=8
• ICMPCODE=0
• ICMPCHECKSUM=6462 -----delete this line
• ICMPIDENT=768
• ICMPSEQ=256
• //ICMPDATA=+++ATH0
• ICMPDATA=0x2B2B2B415448300D
• !Display=Sending some ICMP packets in 2 seconds...
• !Sleep 2000
• !SEND $packets ICMP
• !Display=Packets sent.
• !Sleep 3000

• !Display=$done
• !messagebox=$done
```

- We can't examine the script from the Packet Builder program
- We have to find the file in the Packet Builder directory using explorer.
- The scripts are text files we can edit using Notepad
- Rather than put in the correct checksum value we can just delete the variable from the script.
- This allows Packet Builder to calculate the checksum on the fly for each packet as it sends it out.
- Save and close the file.
- Run the new script

# New script results

```
C:\Documents and Settings\Administrator>netstat -s -p icmp
```

## ICMP Statistics

	Received	Sent
Messages	13	5
Errors	8	0
Destination Unreachable	0	0
Time Exceeded	0	0
Parameter Problems	0	0
Source Quenches	0	0
Redirects	0	0
Echos	5	0
Echo Replies	0	5
Timestamps	0	0
Timestamp Replies	0	0
Address Masks	0	0
Address Mask Replies	0	0

- After the new script is run on the attack PC we have successful pings
- Netstat on the target host shows 5 echoes received and 5 echo replies sent

# CBAC lab revisited

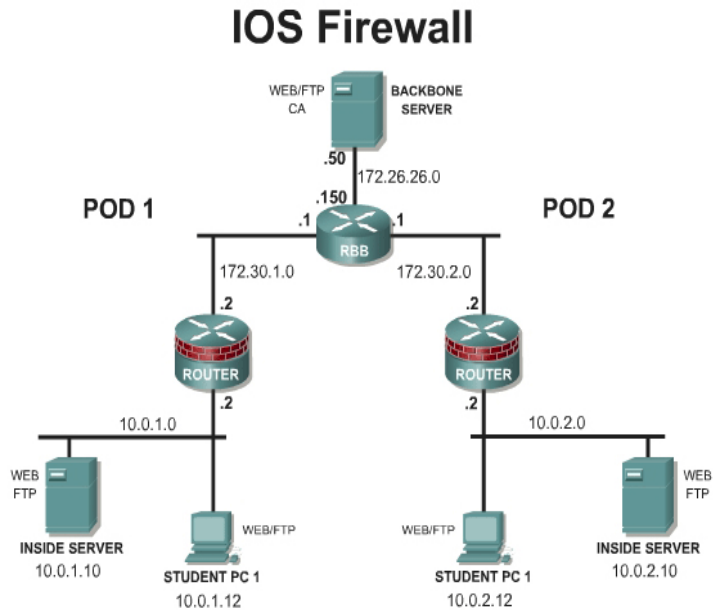
- access-list 100 deny ip 172.30.1.0 0.0.0.255 any
  - access-list 100 deny ip host 255.255.255.255 any
  - access-list 100 deny ip 127.0.0.0 0.255.255.255 any
  - access-list 100 permit ip any any
  - access-list 101 deny ip 10.0.1.0 0.0.0.255 any
  - access-list 101 permit icmp any 10.0.1.0 0.0.0.255 echo-reply
  - access-list 101 permit icmp any host 172.30.1.2 echo-reply
  - access-list 101 permit icmp any host 172.30.1.2 time-exceeded
  - access-list 101 permit icmp any host 172.30.1.2 unreachable
  - access-list 101 permit eigrp any any
  - access-list 101 deny ip 10.0.0.0 0.255.255.255 any
  - access-list 101 deny ip 172.16.0.0 0.15.255.255 any
  - access-list 101 deny ip 192.168.0.0 0.0.255.255 any
  - access-list 101 deny ip 127.0.0.0 0.255.255.255 any
  - access-list 101 deny ip host 255.255.255.255 any
  - access-list 101 deny ip host 0.0.0.0 any
  - access-list 101 deny ip any any log
- Lab Activity: Configure Cisco IOS Firewall CBAC
  - After the router was secured and all the ACLs shown at the left applied to their respective interfaces, the lab instructed us to ping the RBB from the student PC command prompt.
  - This failed until we added the following line to the beginning of ACL 101:
  - access-list 101 permit icmp any 10.0.1.0 0.0.0.255 echo-reply
  - Adding the line permitted us to ping, but did it open a vulnerability in the firewall? We had no way test this since we had no way to generate echo replies.
  - But we can now use Packet Builder to build and send echo reply packets.

# Edit the Packet Builder Script

- // Rafale X script
- // -----
- // Action : Turn off vulnerable modems
- //
- %name=ICMP Echo Request v0.2
- %category=Test
- %date=02/06/2002
- %packetbuildermin=0.2
- %description=Turn off vulnerable modems.
- %description=(ex: Olitec SpeedCom)
  
- // Variables
- \$done=Done !
- \$packets=1000
  
- // Do the stuff...
- IPOFFSET=0
- IPIDENT=9
- IPTTL=255
  
- ICMPATYPE=0
- ICMPCODE=0
- ICMPIDENT=768
- ICMPSEQ=256
- //ICMPDATA=+++ATH0
- ICMPDATA=0x2B2B2B415448300D
- !Display=Sending some ICMP packets in 2 seconds...
- !Sleep 2000
- !SEND \$packets ICMP
- !Display=Packets sent.
- !Sleep 3000
  
- !Display=\$done
- !messagebox=\$done

- Change the ICMPATYPE from 3 to 0. This changes the packets we send from echo requests to echo replies.
- Increase the number of packets to 1000
- Save and close the script

# Reproduce the CBAC lab exercise



- Standard lab setup
- Pod 2 is left as default configuration
- Pod 1 router has the CBAC configuration with the added permit line in ACL 101 to allow echo-replies through to PC1 host. PC1 host is the target host.
- DMZ 2 PC with Packet Builder installed is the attack PC
- Run the script to see if the packets get through to PC 1

# Target host hacked

```
C:\Documents and Settings\Administrator>netstat -s -p icmp
```

## ICMP Statistics

	Received	Sent
Messages	23	14
Errors	8	0
Destination Unreachable	0	0
Time Exceeded	0	0
Parameter Problems	0	0
Source Quenches	0	0
Redirects	0	0
Echos	10	4
Echo Replies	5	10
Timestamps	0	0
Timestamp Replies	0	0
Address Masks	0	0
Address Mask Replies	0	0

```
C:\Documents and Settings\Administrator>netstat -s -p icmp
```

## ICMP Statistics

	Received	Sent
Messages	1023	14
Errors	8	0
Destination Unreachable	0	0
Time Exceeded	0	0
Parameter Problems	0	0
Source Quenches	0	0
Redirects	0	0
Echos	10	4
Echo Replies	1005	10
Timestamps	0	0
Timestamp Replies	0	0
Address Masks	0	0
Address Mask Replies	0	0

```
C:\Documents and Settings\Administrator>
```

- Echo replies received: 1005, all 1000 packets got through!

# Conclusions

- When the added ACL line is removed and the script is run again the packets do not get through.
- The mistake in the lab, I believe, is to ask us to ping RBB from the host command line. It is not wise to allow inside hosts to ping outside the protected network. The ping should have been done from the Pod router (from hyper-terminal).
- Could this vulnerability be used in a distributed denial of service attack? Distributed Denial of Service attacks TFN (Tribe Flood Network), TFN2K, and Stacheldraht (German for "barbed wire") all use echo-reply packet to coordinate and control their attacks.
- Packet Builder, when used with Ethereal, is a handy tool to test for security vulnerabilities, unfortunately it's also a handy hacker tool to exploit those vulnerabilities.